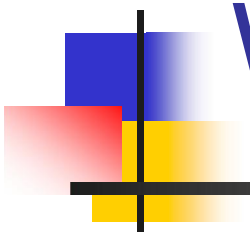
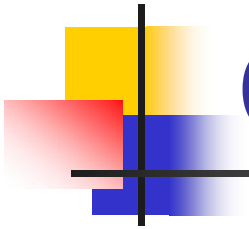


Web Based Applications

Dr. Mohammad Iqbal

Thanks to Andy Wood

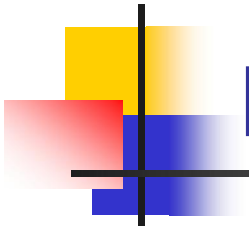


Overview

- Definisi Aplikasi berbasiskan web
- Kelebihan Aplikasi berbasiskan web
- Kekurangan Aplikasi berbasiskan web
- Arsitektur Dasar Aplikasi berbasiskan web
- Teknologi Implementasi Aplikasi berbasiskan web
- Bagaimana menambah interface web pada aplikasi *native*?
- Desain yg dipertimbangkan untuk Aplikasi berbasiskan Web (as opposed to normal apps)
- Adakah alternatif untuk aplikasi berbasiskan web?

Definisi Aplikasi Berbasiskan Web

- *"A web based application is a software package that can be accessed through the web browser. The software and database reside on a central server rather than being installed on the desktop system and is accessed over a network."* (NetSity corporate homepage)
- A common example is web-based email. ie, Hotmail, Gmail, Yahoo! mail or Gunadarma's web mail.

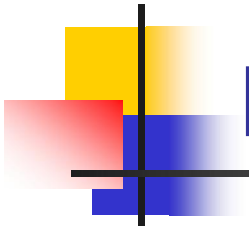


Kelebihan Aplikasi berbasiskan web

- App runs server side (CPU, disk space, configuration)
- No install, packaging, CDs, upgrades, configurations or tweaking of settings on the client side.
- Greater responsibilities and control placed in the hands of the system administrators (as opposed to the users)
- Data is likely more secure (stored server side, w/ proper security measures and backup)
- Machine independent (any user can log in from any computer.

Kelebihan Aplikasi berbasiskan web
(lanjutan)

- One application will run on any and all platforms, assuming standards compliant code and browsers.
- Reduced external network traffic (ex. Database heavy applications)
- Lower client side system requirements (machine only needs network access and the ability to run a compliant web browser)



Kekurangan Aplikasi berbasiskan web

- "Who has my data?" Essentially, not you.
- Issues of trust; many users do not trust other people, even within the same company, to keep their data safe and secure.
- Response time. While the actual execution of the app may be much quicker, user response time can be noticeably slower than a local app.
- Internet (or network) connectivity is not (yet) ubiquitous.

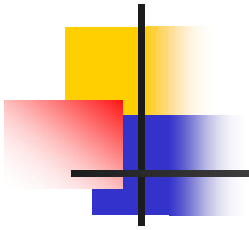
Kekurangan Aplikasi berbasiskan web
(lanjutan)

- Browser compatibility can still be a problem.
- Some tasks that are simple in traditional application development, are quite complicated from a web application (ex, local printing)
- Security concerns limit what you can accomplish (limited access to the users local machine).
- Depending on the application, usability can be very bandwidth sensitive.

Kelebihan Aplikasi berbasiskan web
(lanjutan)

Some advantages are also disadvantages:

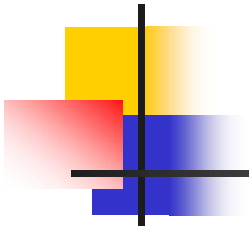
- Running applications server-side requires servers with sufficient power (CPU, memory, disk, bandwidth) to handle multiple simultaneous users.
- Placing greater control in the hands of the system administrators is only an advantage with a sufficient number of competent admins.



The Right Tool untuk *the Right Job*

Web based applications are ideal for:

- Database heavy applications
- Applications that must be used remotely
- Varied user base
- Low GUI requirements
- Not performance sensitive (client side)

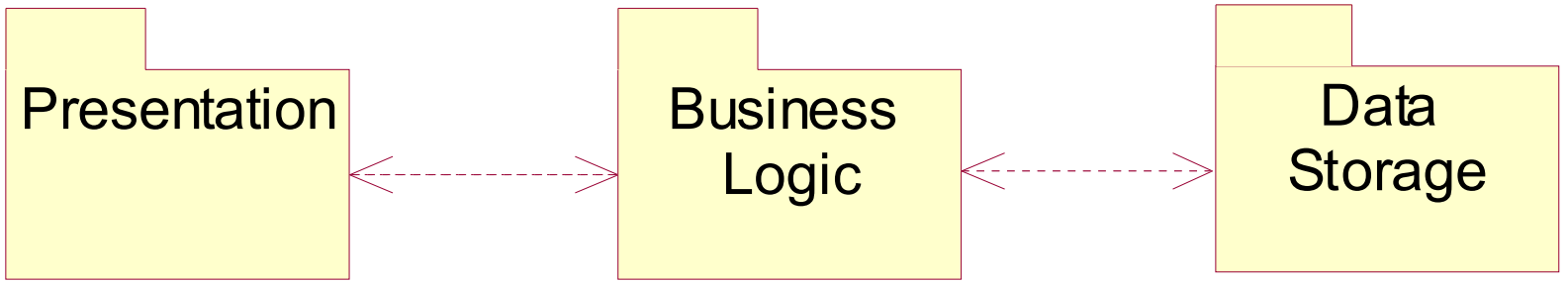


The Right Tool untuk *the Right Job*

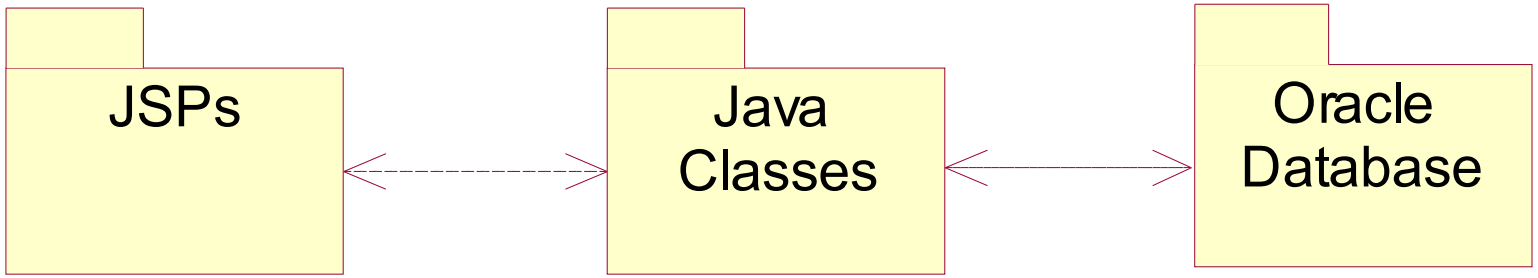
Web based applications may not work for:

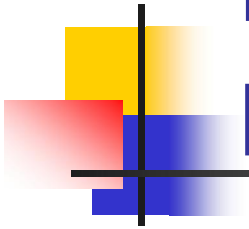
- GUI intensive applications
- Applications requiring access to users local machine
- Performance sensitive applications
- Situations with unreliable or limited network connectivity

Arsitektur Dasar Apl berbasiskan web



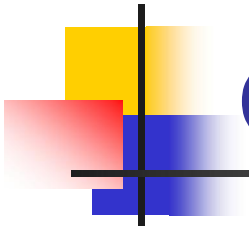
Example of a possible J2EE implementation





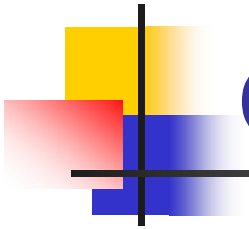
Bagaimana menambah interface web pada aplikasi *native*?

- Can be extremely difficult (if not impossible) or trivially easy, depending on the application.
- Can be done at different levels:
  - New program, share data store (Webmail).
  - Old program, with a new, web based, GUI.



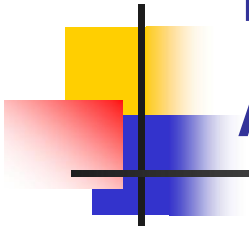
Contoh Java Class

```
public class User {
        …
        public boolean hasEditRights() {
                return editRights;
        }
        …
}
```



Contoh JSP Code

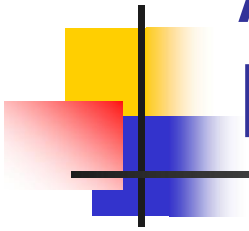
```
…
<% if (user.hasEditRights()) { %>
        <input name="test" value="Text Here">
<% } else { %>
        <input name="test" value="Text Here" readonly>
<% } %>
…
```



Desain yg dipertimbangkan untuk Aplikasi berbasiskan Web

- "back-end" code can generally use standard object-orientated programming practices and standards
- "presentation" code differs somewhat because of the restrictions imposed by web standards.
- Same guidelines for good GUI design apply, but additional considerations must be made.
- Proper web site design considerations also apply
- The application code must be built to handle multiple simultaneous users.

Adakah alternatif untuk aplikasi berbasiskan web?



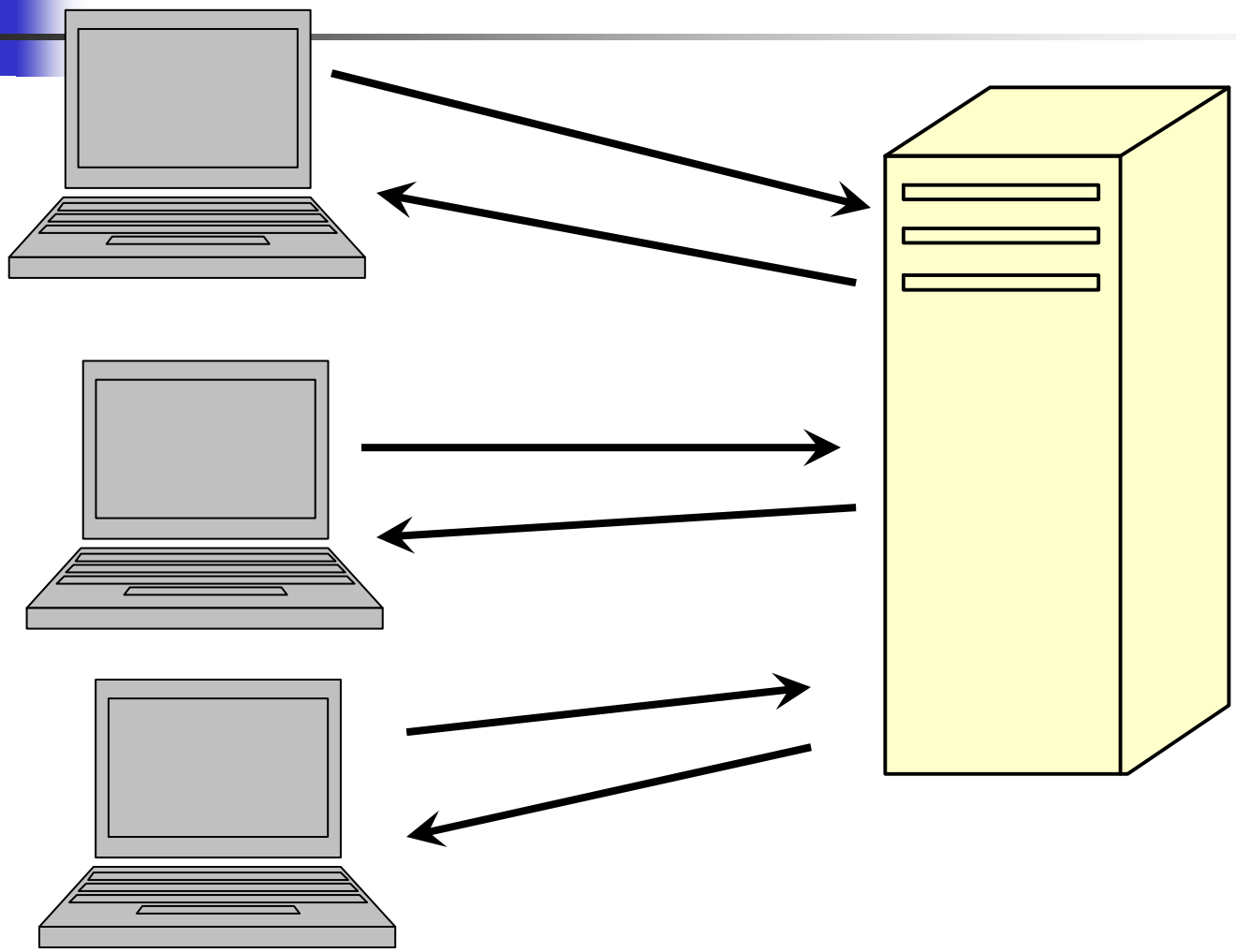
- Citrix
- Remote Desktop
- Hybrid Application
- Others

Web Application Architecture

Dr. Mohammad Iqbal

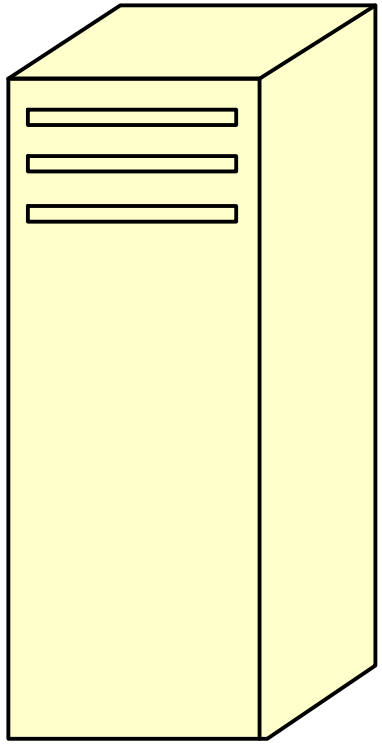
Source : Bird Book pp 1-14

Client Server Model

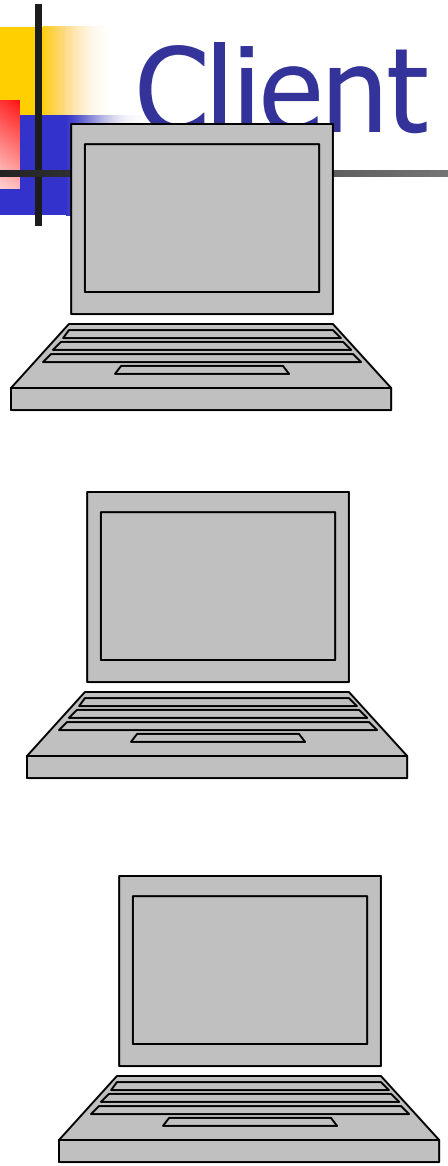


Server Applications (Software)

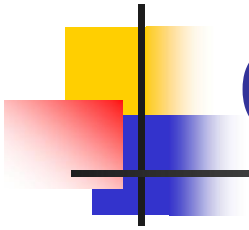
- Management and maintenance of Data including
  - User login data
  - Application data
- Data processing
- Centralized
- Access via Login



Client Applications (Software)

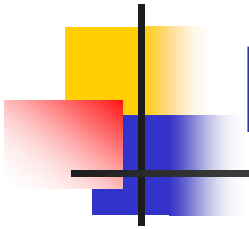


- Provides user interface
- Stores some settings
- Can do some data processing
- Little to no application data storage
  - Same view of data no matter where you login



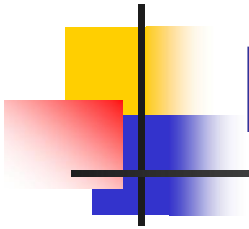
Client-Server Advantages

- Centralized Data Storage
- No data redundancy (no duplication of data)
- Reduces data dependencies
  - If data is stored on each user's system and each system is different than data depends on how the user system is designed
  - Data can not be shared easily if such dependencies exist



Classic Example: Early Banking Systems

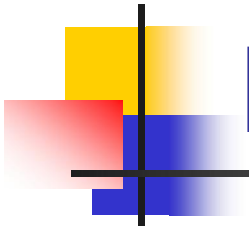
- **Network**: Local Area Network (LAN) covering local office branch.
- **Server**: Mainframe-like server "in the back" running custom banking system
- **Client**: Windows PC with client interface for each bank teller.
- Data is the same no matter what teller you go to.
- Data is NOT the same if you go to another branch unless servers exchanged some data at night.



Classic Example: Early Banking Systems

The Obvious Future:

- Change the LAN to a wide area network covering all the branches.
- Get rid of the individual servers at each branch
- Have clients connect to central server where ALL the banking data is stored.

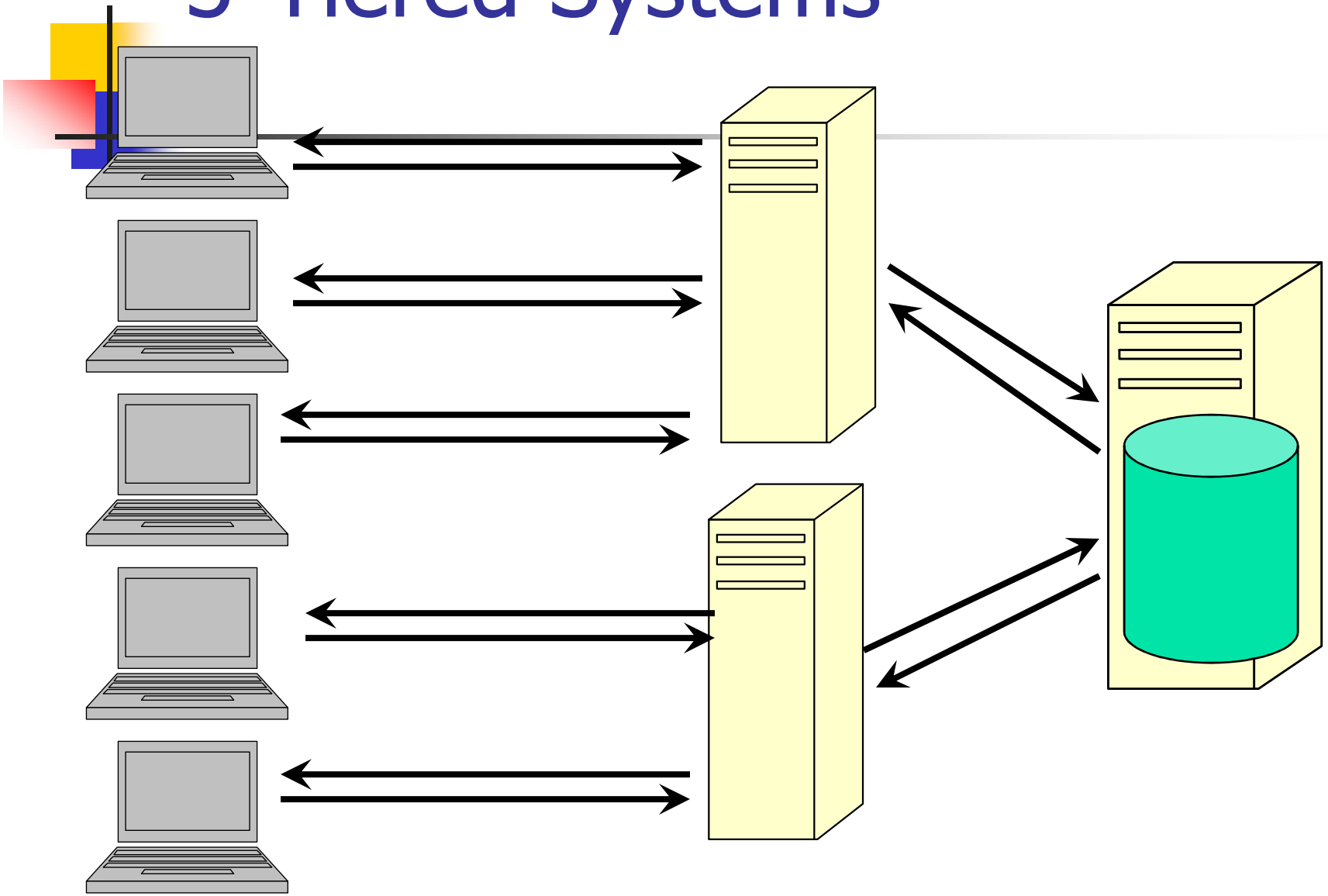


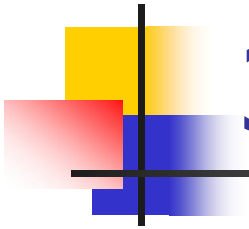
Classic Example: Early Banking Systems

The Obvious Problems:

- Large banks could have thousands of tellers connecting to the central server.
- Combining data from all branches requires severs with lots of storage capacity.
- Branch data could be stored in different formats.
- Lack of Standardization.

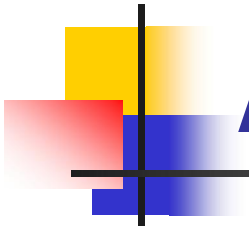
3-Tiered Systems





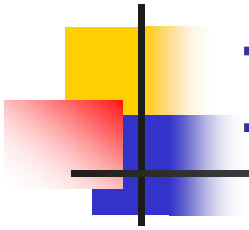
3-Tiered System

- **Database Tier (Database Server)**
  - Data storage and low level data manipulation
- **Server Tier (Application Server)**
  - Manage client connections and data processing
- **Client Tier (Client Software installed locally)**
  - User interface and some data processing



Advantage of 3-Tier Systems

- Central Database Server accessed by multiple Application Servers
- In turn, each Application Server could independently manage thousands of users
- **Database Server** is specially designed to do its job
  - Database Operations: Update, Insert, Remove, etc.
  - Lots of disk storage and memory needed
- **Application Servers** can be added to support more users or **DIFFERENT APPLICATIONS**
  - Server Operations: Complex application-dependent computations



Internet vs. WWW

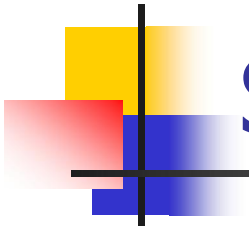
Internet is the infrastructure that makes the WWW work.

- **Packet Switching**
- **TCP/IP Protocol**
- **Physical Infrastructure**
  - Fiber-optics lines, wires
  - Satellites, Cable Modems
  - Routers, Hubs, Network Cards, WiFi systems, etc.

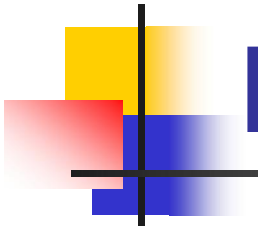
WWW is just one of many "virtual networks" built on the Internet.

- **Websites**: http, https, etc.
- **Email**: pop, imap, etc.
- Other systems: ftp, instant messaging, etc.
- **Note**: Even to this day companies have "private virtual networks" that use the Internet, but are proprietary, locked-down.

WWW – Ultimate Client-Server System

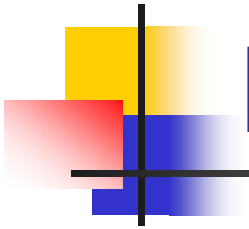


- Already Standardized
- Built on the Widest Area Network you could imagine, i.e., **The Internet**
- Standardized Clients that are free to use
  - IE, Firefox, Safari, etc.
- Lots of Servers already in place
  - Apache, Windows Server (IIS), etc.
- **Database Servers**
  - **Umm, this was initially missing**



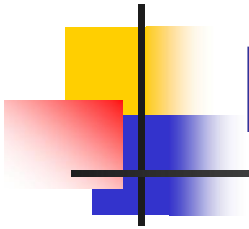
First Web Applications

- 1993 – Rob McCool proposed a framework called
  - **CGI (Common Gateway Interface)**
- Data passed from a web browser to the server
  - **GET** - passed via URL variables
  - **POST** - passed via HTML forms
- Web server daemon (httpd) could then make remote system calls
- Example
  - Web server could run a C++ program and write the output to public HTML folder
  - Web server would send response back with location of the output.



First Web Applications

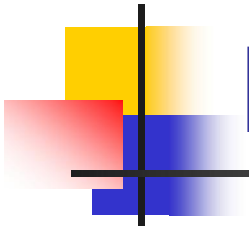
- Using CGI, web server could run
  - C++ programs
  - Perl Programs
  - Fortran Programs
- C++ has library functions that allow you to connect to a number of different databases:
  - Oracle
  - Sybase
  - DB2



First Web Applications

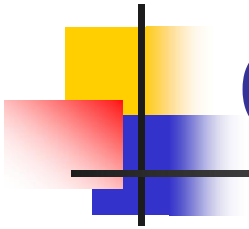
Problem:

- To develop web applications you need to know
  - Exactly how your server is configured
  - HTML forms
  - GET and POST conventions
  - C++ database libraries
  - SQL language
- Getting all these things to work together is a pain in the ***.



First Major Improvement

- 1995 – JJ Allaire developed "a hack" that allowed a web servers to communicate with other systems, namely a database system.
- **Key:**
  - Instead of using "a middle-man" C++, Perl, Java, etc.
  - Developer could directly add **code** to the their web pages
  - Using a special Markup Language, this **code** could be embedded in any web page.
  - Worked seamlessly with HTML forms
  - Server process **code** directly

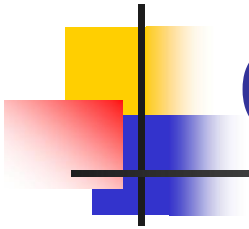


ColdFusion

- JJ Allaire went on to form a company **Allaire** which developed his idea into a product called **ColdFusion**
  - ColdFusion Markup Language (**CFML**)
  - ColdFusion Server (addon to popular Web Servers like Apache, Microsoft's IIS).
- Notes:
  - Allaire was bought by Macromedia 2001.
  - Macromedia was bought by Adobe in 2005.

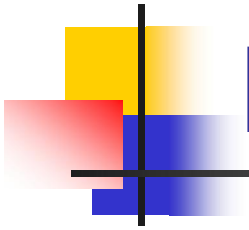
ColdFusion Example

```
<cfset droplist = "colorlight,colordark">
<cfoutput>
 <form action="#cgi.script_name#" method="get" name="choosecolors"
   id="choosecolors">
 <fieldset>
   <legend>Customize Site</legend>
   <label for="colorlight">light color</label>
   <input type="text" name="colorlight" id="colorlight" size="10"
    value="#url.colorlight#" /><br />
   <label for="colordark">dark color</label>
   <input type="text" name="colordark" id="colordark" size="10"
    value="#url.colordark#" />

   #getTokensMinusArg('inputs',droplist)#

   <input type="submit" name="changecolors" value="Reload" />
 </fieldset>
 </form>
</cfoutput>
```



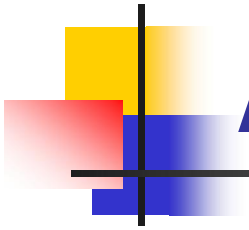
ColdFusion

- The term Cold Fusion refers to a nuclear reaction that can occur at room temperature
- In the 1980's it was believed that Cold Fusion was a physical possibility.
- If Cold Fusion could be achieved then almost unlimited power could be generated by a reaction you could perform in your own kitchen.
- JJ Allaire aptly named his product ColdFusion because he believed it would be as great as real "Cold Fusion."
- He was right, but…



Problems with ColdFusion

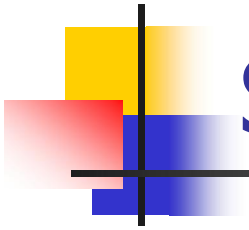
- The concept is great, but the implementation sucked.
- CFML is difficult to learn, overly complex
- ColdFusion server was initially very slow
- You have to pay for it
- Initially, it only worked for Microsoft's server, which you have to pay for.
- Made JJ Allaire a multimillioniare but…



Alternative's to ColdFusion

- Microsoft developed it's own system called Active Server Pages (ASP), which was more tightly integrated with their web server.
- Sun Microsystems, developed Java Server Pages (JSP), which worked better with its server and used Java as the application language.
- The Apache Community spawned PHP which is as good as the systems above, but Open Source.

WWW – Ultimate Client-Server System



- Already Standardized
- Built on the Widest Area Network you could imagine, i.e., **The Internet**
- Standardized Clients that are free to use
  - IE, Firefox, Safari, etc.
- Lots of Servers already in place
  - Apache, Windows Server (IIS), etc.
- **Database Servers**
  - **ColdFusion, ASP, JSP, and PHP all have built-in support to connect to databases.**

Questions?