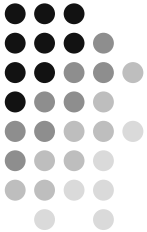
N-tier Architectures



In this section…

- Iadh's been blasting you with SQL to a high standard!
- Look at applications of SQL in programming languages, in particular
  - JDBC, SQLj
  - PHP
- Also other crucial aspects of data intensive systems
  - transactions
- First, though, in this lecture
  - attempt to fit all this in context – Enterprise computing

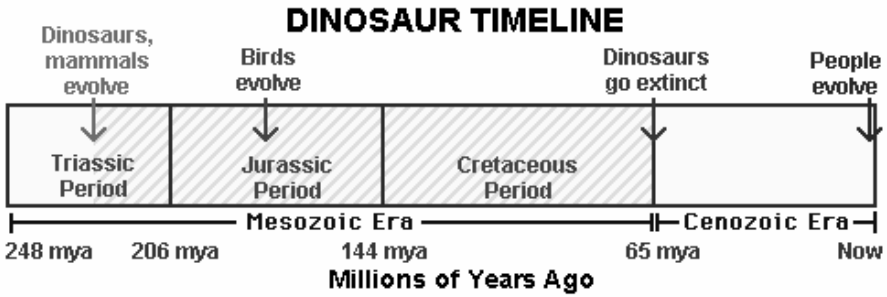
What characteristics do we want our Software to have (PSD…?) ?

- Easily Developed
- Speedily Developed
- Agile
- Easily Understood
- Easily Maintained
- Easily Modifiable
- Reliable

Major components in Enterprise computing

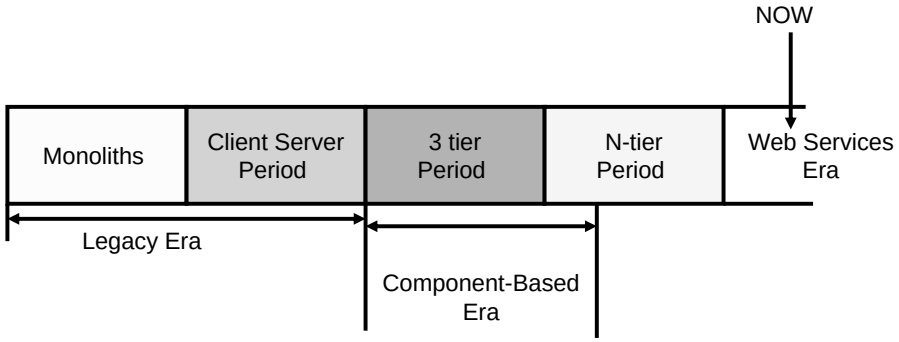
- Basically
  - data
  - business logic
  - interfaces
- But this can be/is implemented in more layers often.

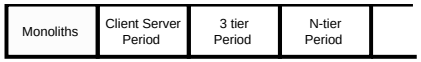
The Mesozoic Era



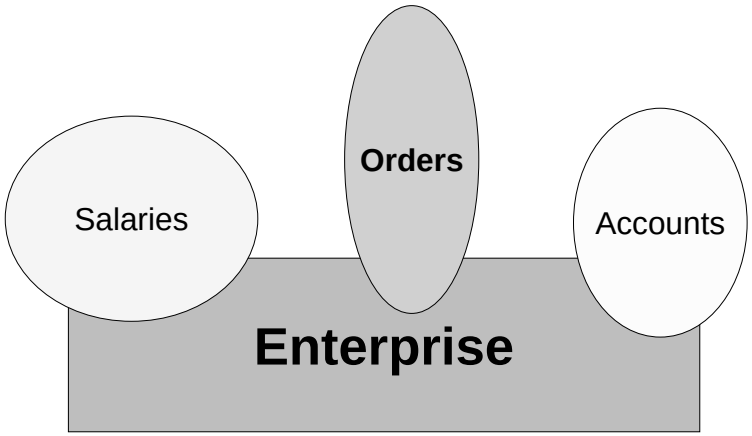
The dinosaurs and the mammals appeared during the Triassic period, roughly 225 million years ago. The dinosaurs went extinct 65 million years ago

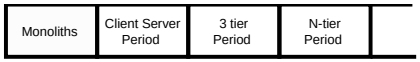
Computer Systems Timeline





In the beginning….





Monolithic Era

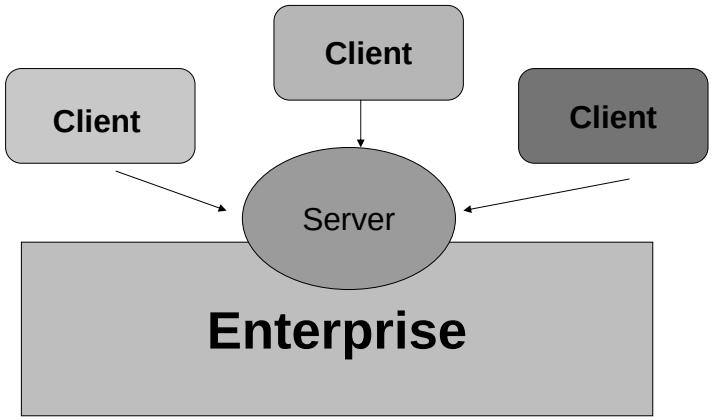
- Single chunks of code
- Data Storage embedded
- UI embedded
- System rules embedded
- No inter-system dependencies

Monolithic Era – reasons for *existence*

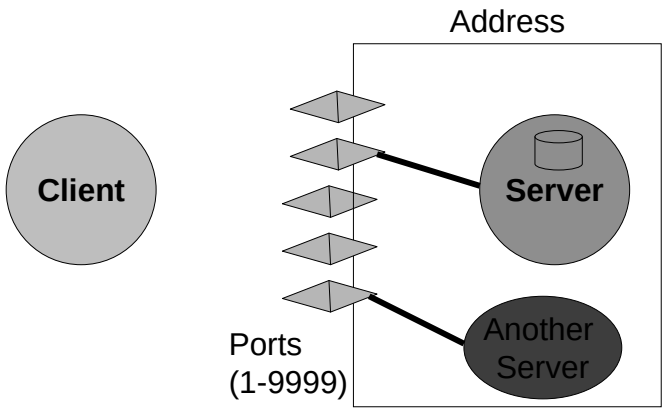
- Mainframe platforms
- Few or no tools existed
- Off-the-shelf databases did not yet exist
- Simple and easily understood
- Natural way of implementing many business processes

Monolithic Era – reasons for *extinction*

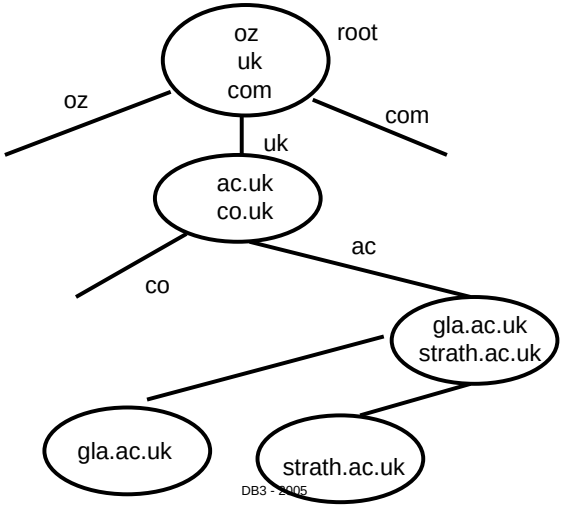
Evolution – 1970s Client Server Era



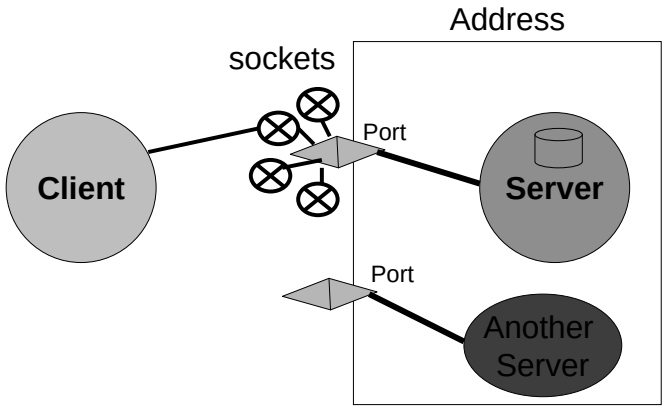
Client - Server



Addressing – the DNS

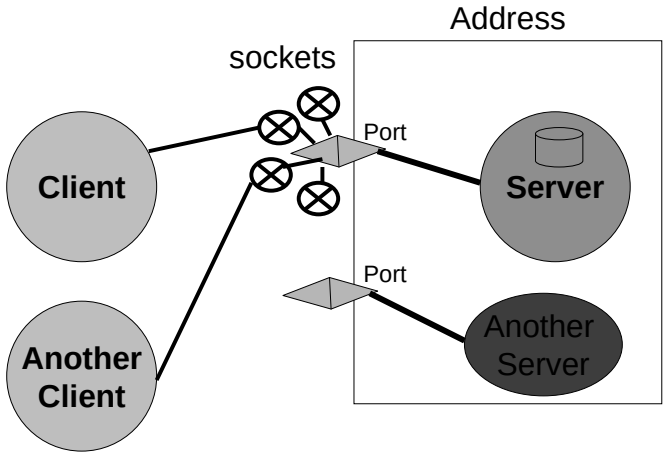


Client - Server



Server Protocol

Client - Server



Server Protocol

Client-Server Era - Reasons

- Advent of PCs
- Availability of off-the-shelf DBMSs
- Advent of RAD Tools
- Acceptable networking facilities
- Distributed Processing load across clients
- Visualisation at the client level and data processing at the server level

Whence the business logic?

- At the client?
  - Yes, usually – hence *Fat Client*
  - Difficult to deal with maintenance and change
- At the server?
  - Could store functions in the DBMS but it overloads the server
- The client and server are tightly coupled.

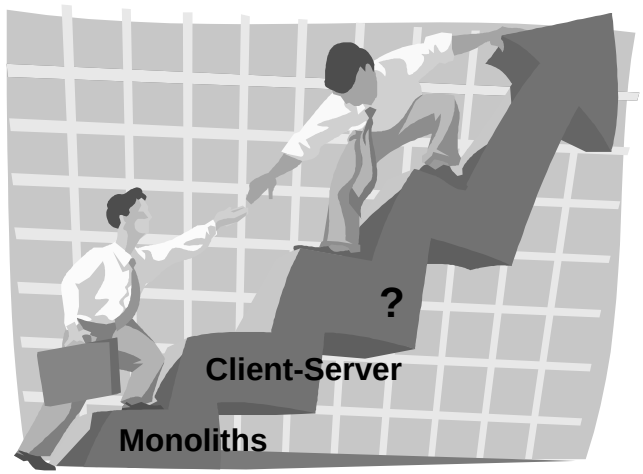
Client-Server deficiencies

- Difficult to reuse client with different data sources
- Difficult to reuse server on different front-end
- Development tools were poor
- Developers were contaminated by the monolithic culture

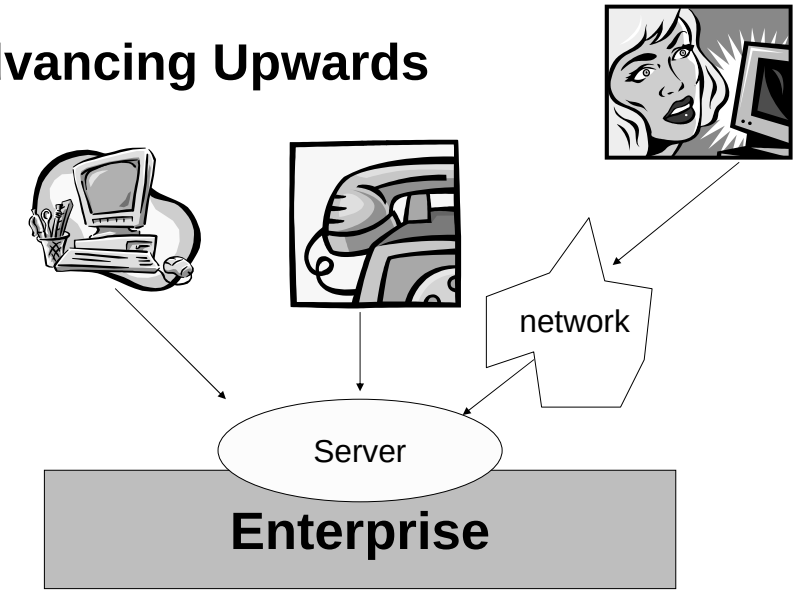
Developers' myopia

- No concept of separating out business logic
- Focus on reuse OF CODE
- The first generation tools were impressive to people who had had **no** tools

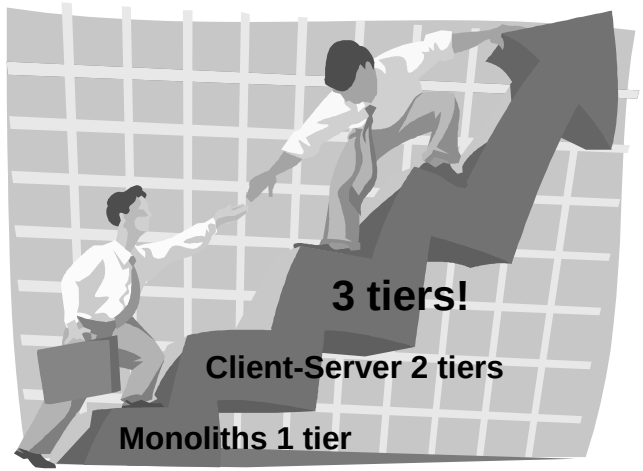
Next step?



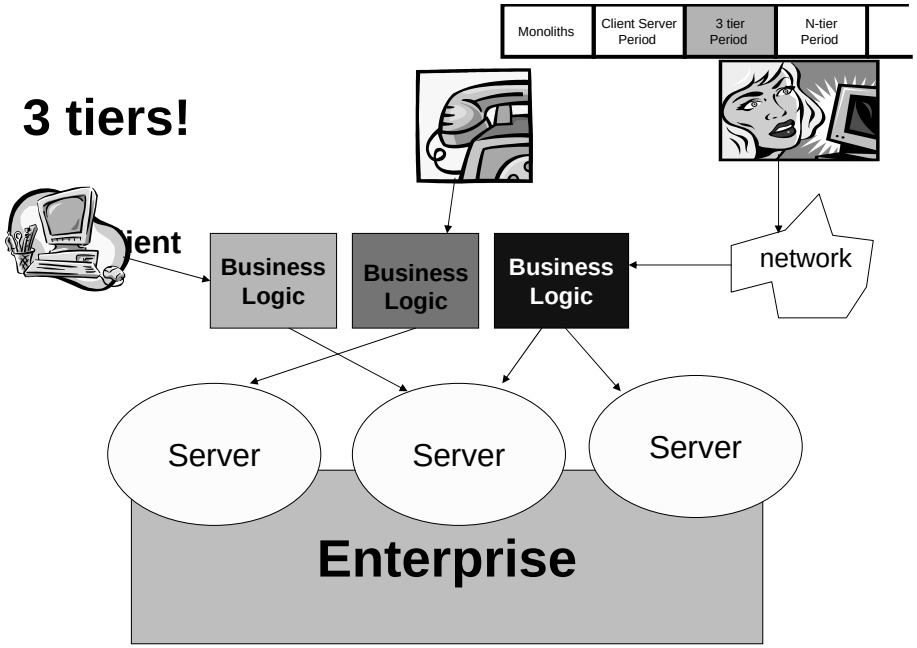
Advancing Upwards



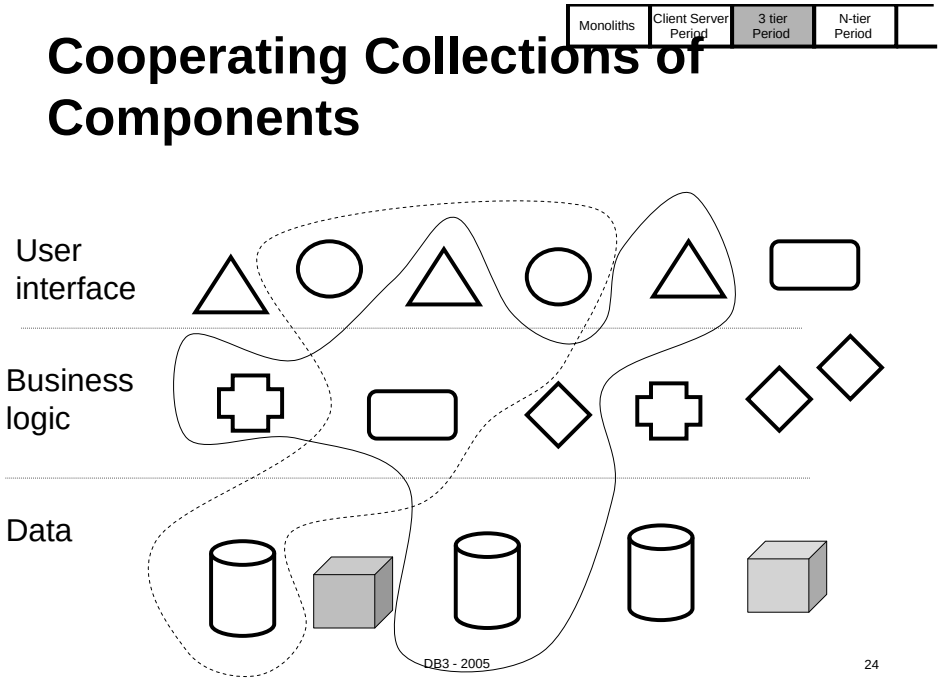
1990s – what next?



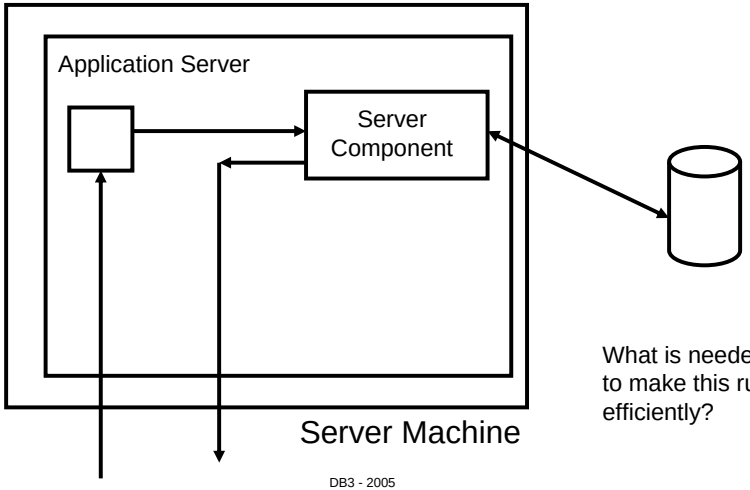
3 tiers!



Cooperating Collections of Components



Server-Side Processing



What is needed to make this run efficiently?

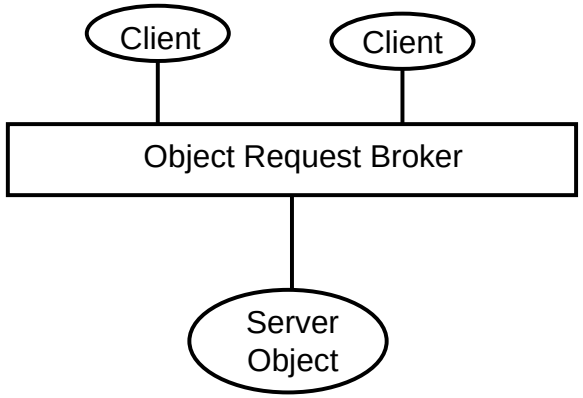
Advantages of 3 tier

- Loosely coupled
- Agile Software
- Use of pre-assembled parts
- Promotes reuse of parts
- Eases maintenance
- Ease of versioning

Interconnection?

- Customised connection would be too difficult
- Use standardised data bus like CORBA
- So MxN connection is simplified to 1x1 connection

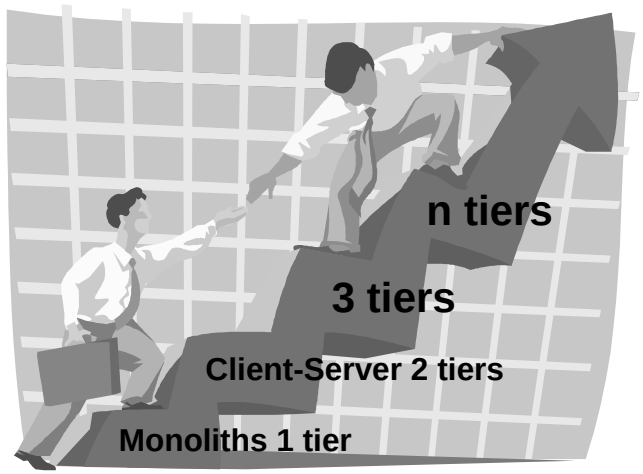
Data Bus Concept



Limitations

- Technical
  - Size of executable
  - Speed of execution
- Cultural
  - Updating legacy systems to 3 tier is not always acceptable to people used to monoliths

What next?

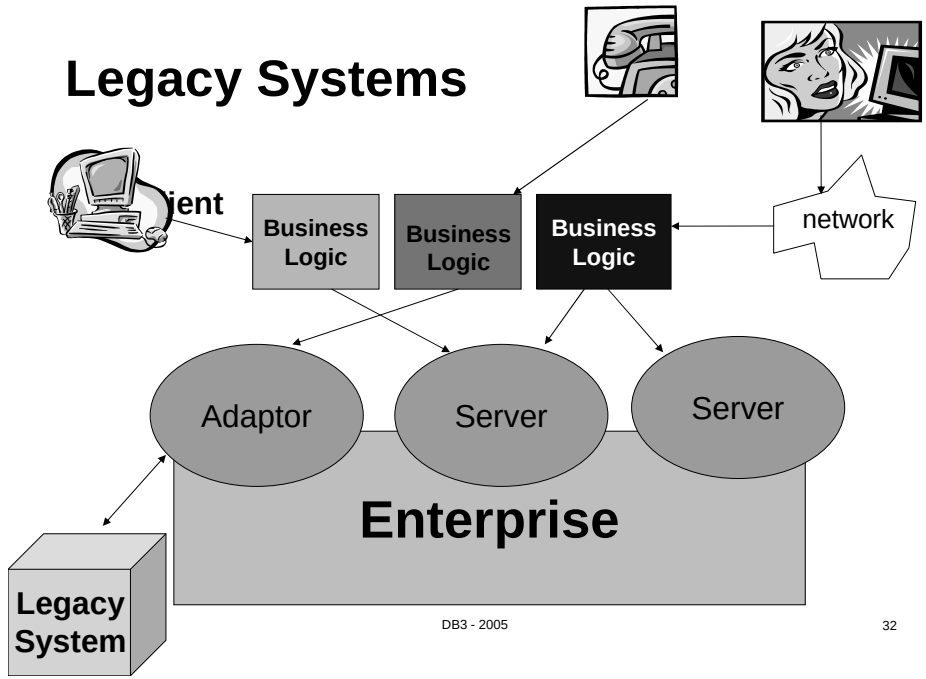


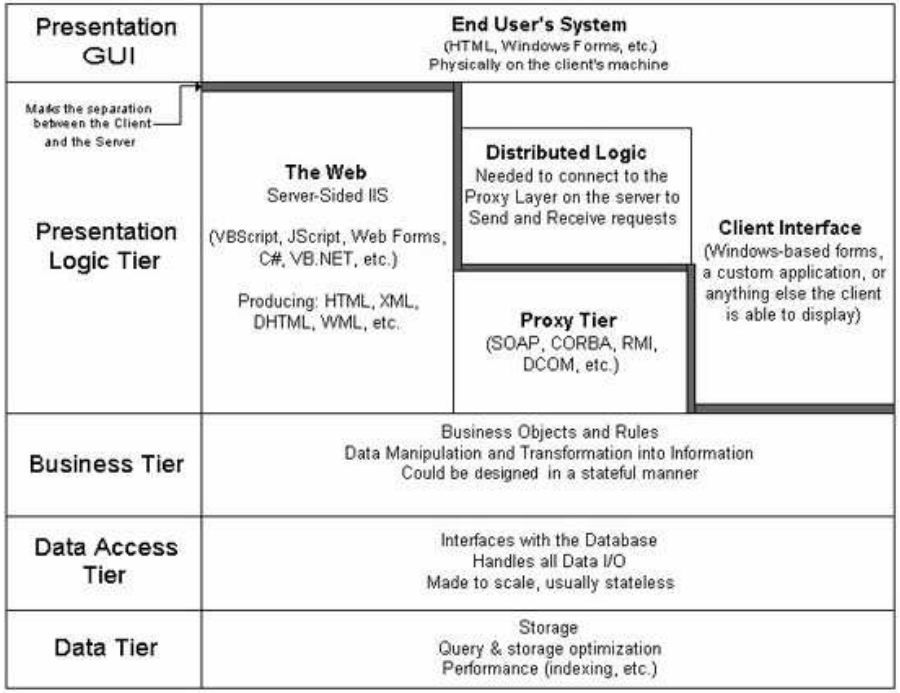
4 tiers…

4	Client
3	Web Server
2	Business Logic
1	Data

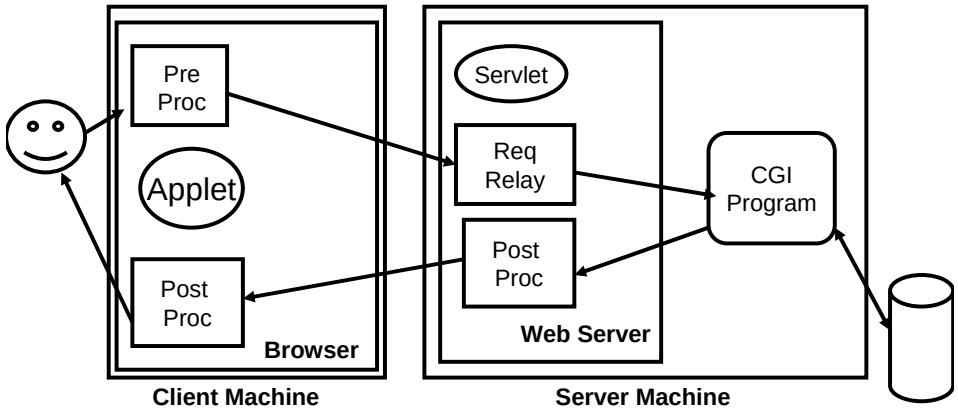
Why is this interesting?
Why necessary?
What's not enough with 3-tier?

Legacy Systems

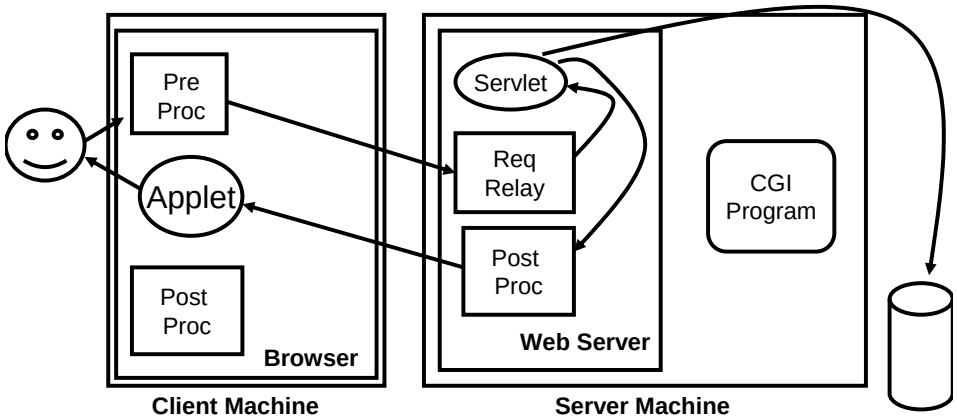




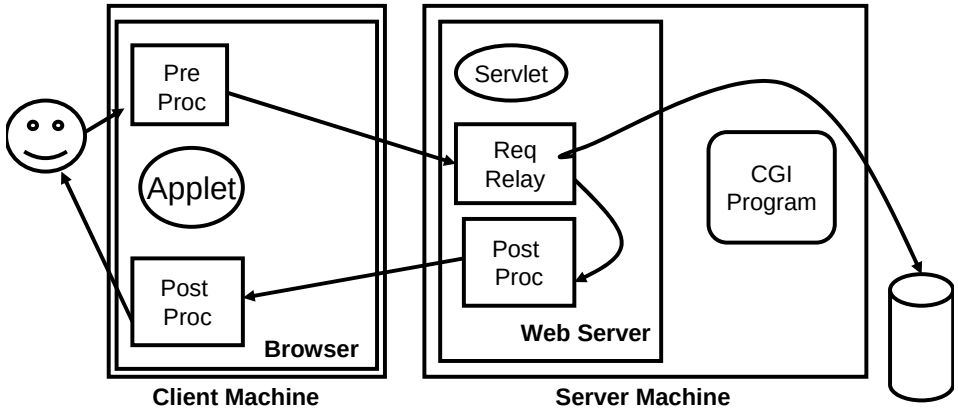
N-tier architecture



N-tier architecture



N-tier architecture



Crucially…

- Software architectures follow advances in systems architectures
- Note ***follow***
  - Each new sys arch leads to a right mess initially –
    - exciting new functionality gained at terrible price
  - until enough experience
    - mindset changes among developers
    - development of truly innovative software structures
- Software architectures – an ongoing rsch area