Introduction to J2EE

The Java Programming Language Platforms

There are three platforms of the Java programming language:

- Java Platform, Standard Edition (Java SE)
- Java Platform, Enterprise Edition (Java EE)
- Java Platform, Micro Edition (Java ME)

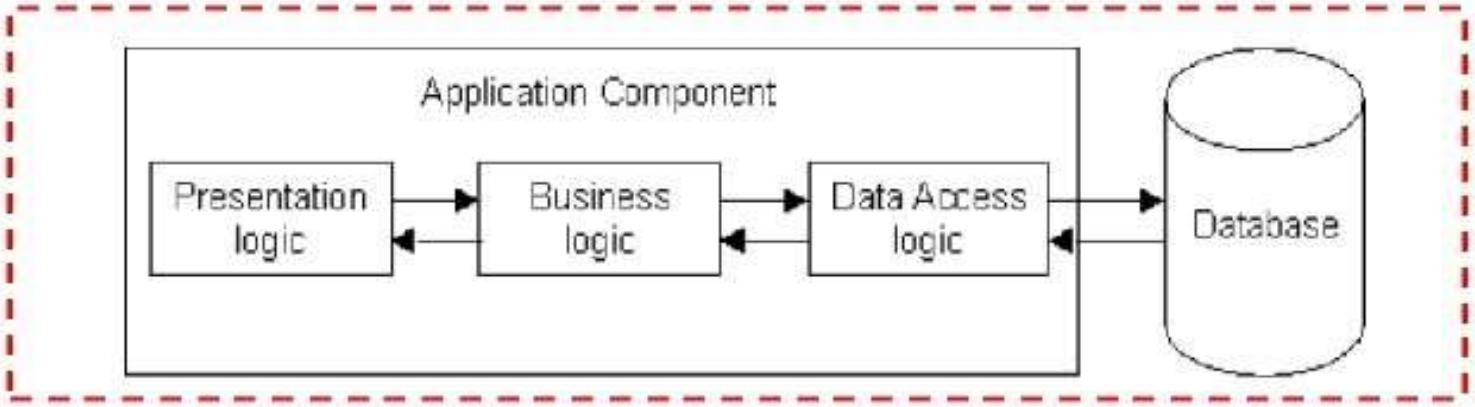
J2SE	J2ME	J2EE
Java 2 Standard Edition	**Java 2 Micro Edition**	**Java 2 Enterprise Edition**
Java standard edition is use to develop client-side standalone applications or applets	Java micro edition is use to develop applications for mobile devices such as cell phones	Java enterprise edition is use to develop server-side applications such as Java servlets and Java Server Pages

Distributed Multi-tiered Applications

- The J2EE platform uses a multi-tiered distributed application model for both enterprise applications
- Application logic is divided into “components” according to function, and the various application components that make up a J2EE application are installed on different machines depending on the tier in the multi-tiered J2EE environment to which the application component belongs

- The J2EE technologies can be broadly classified into four different categories:
  - Client-side technologies
  - Component technologies
  - Service technologies
  - Communication technologies
- Component technologies include:
  - Servlets
  - Java Server Pages
  - Enterprise JavaBeans
    - Session Beans
    - Entity Beans
- Service Technologies include:
  - Java Database Connectivity
  - Java Transaction API and Service

1-Tier Architecture



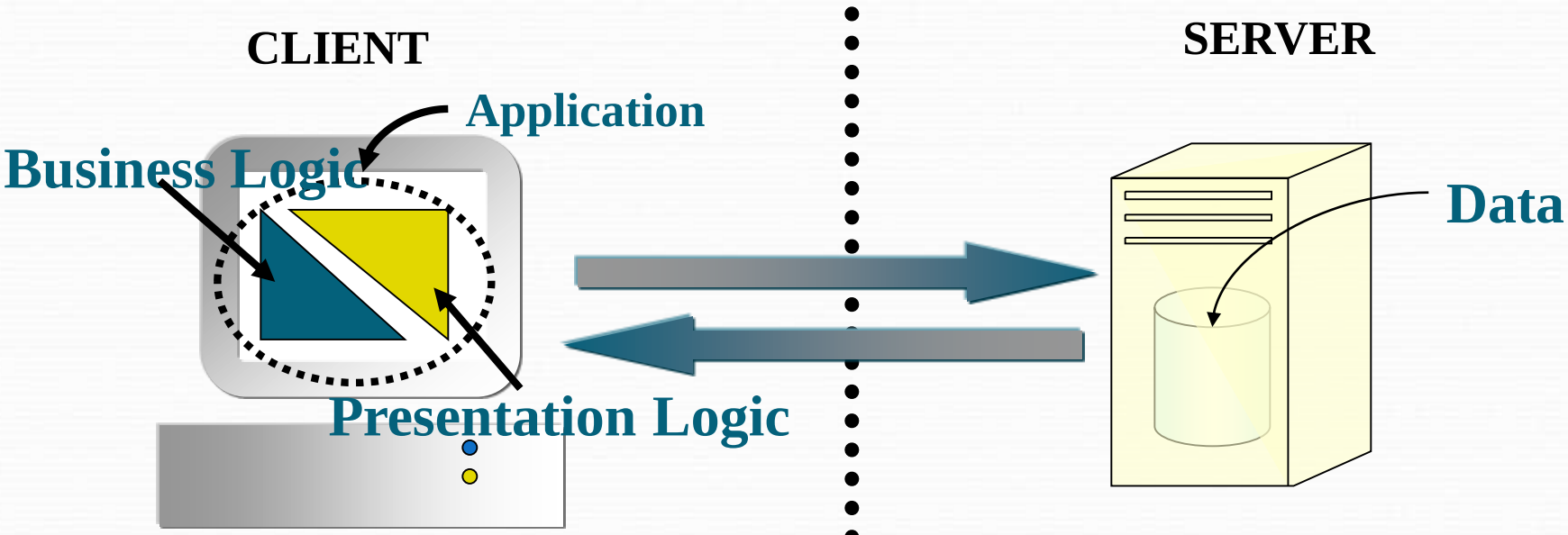
All 3 layers are on the same machine

- All code and processing kept on a single machine

Presentation, Logic, Data layers are tightly connected

- Scalability: Single processor means hard to increase volume of processing
- Portability: Moving to a new machine may mean rewriting everything
- Maintenance: Changing one layer requires changing other layers

Client-Server Architecture



Clear division between **Application** and **Data**

Entire processing load on client

Server acts as traffic controller

2-Tier Architecture

Client-Server: The Drawbacks

Business logic present on each client

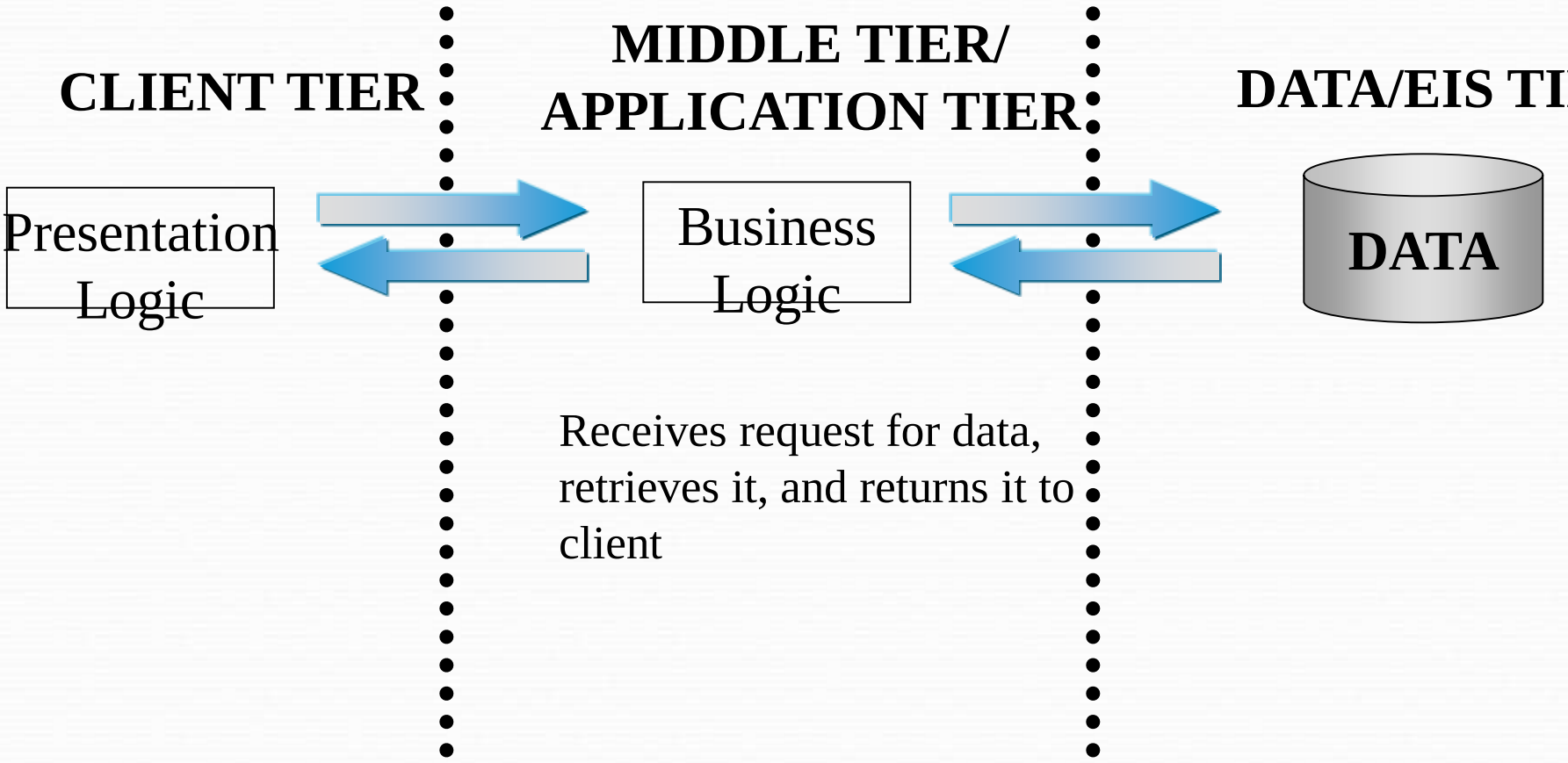
Client waits longer for response

Business logic + presentation logic bundled together-therefore scalability problems

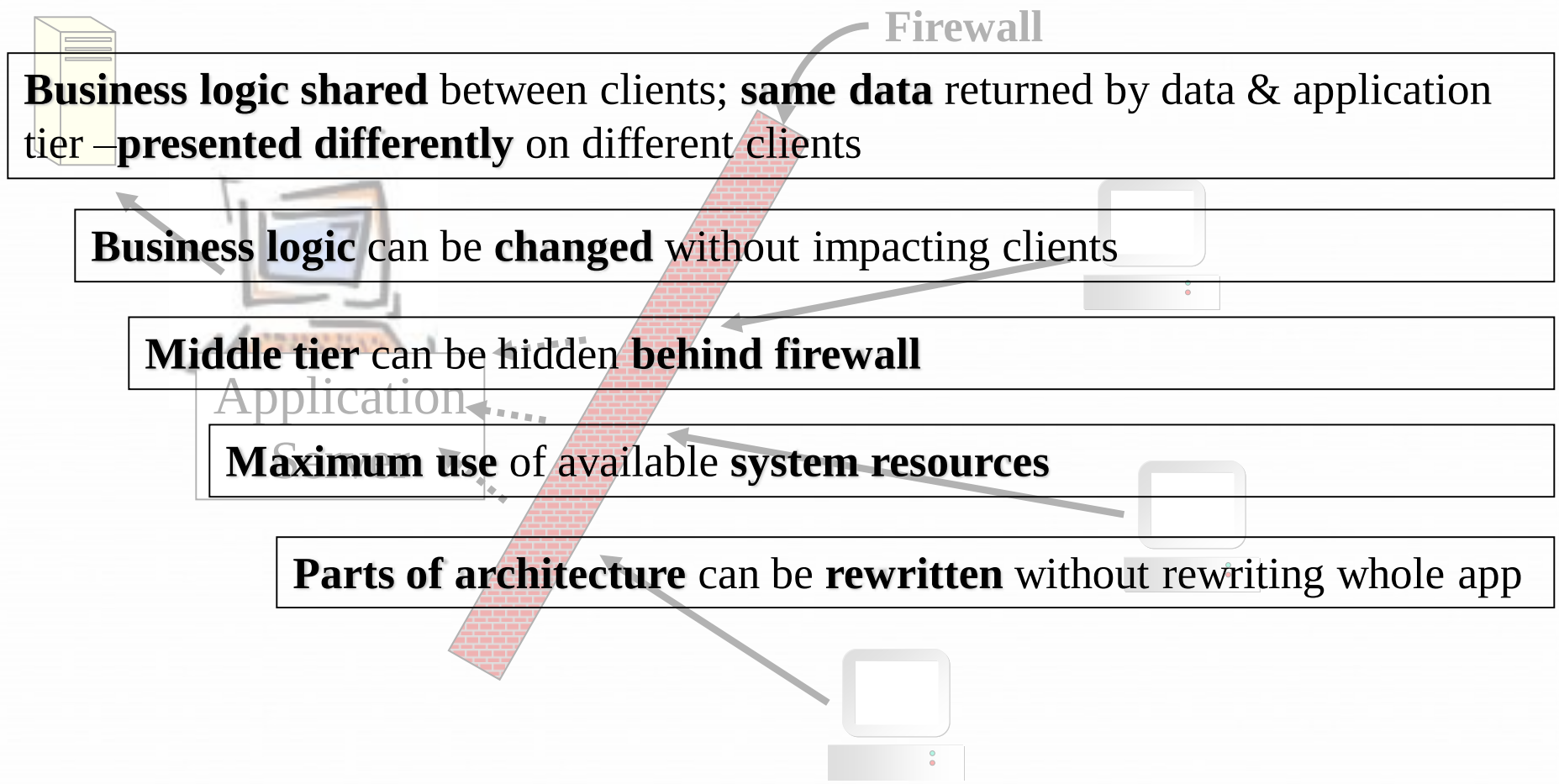
Load on server and network as all clients send request to 1 server

Tiny change to application— entire application has to be changed, and the clients upgraded

3-Tier Architecture



3-Tier Architecture: the benefits



Business logic shared between clients; **same data** returned by data & application tier –**presented differently** on different clients

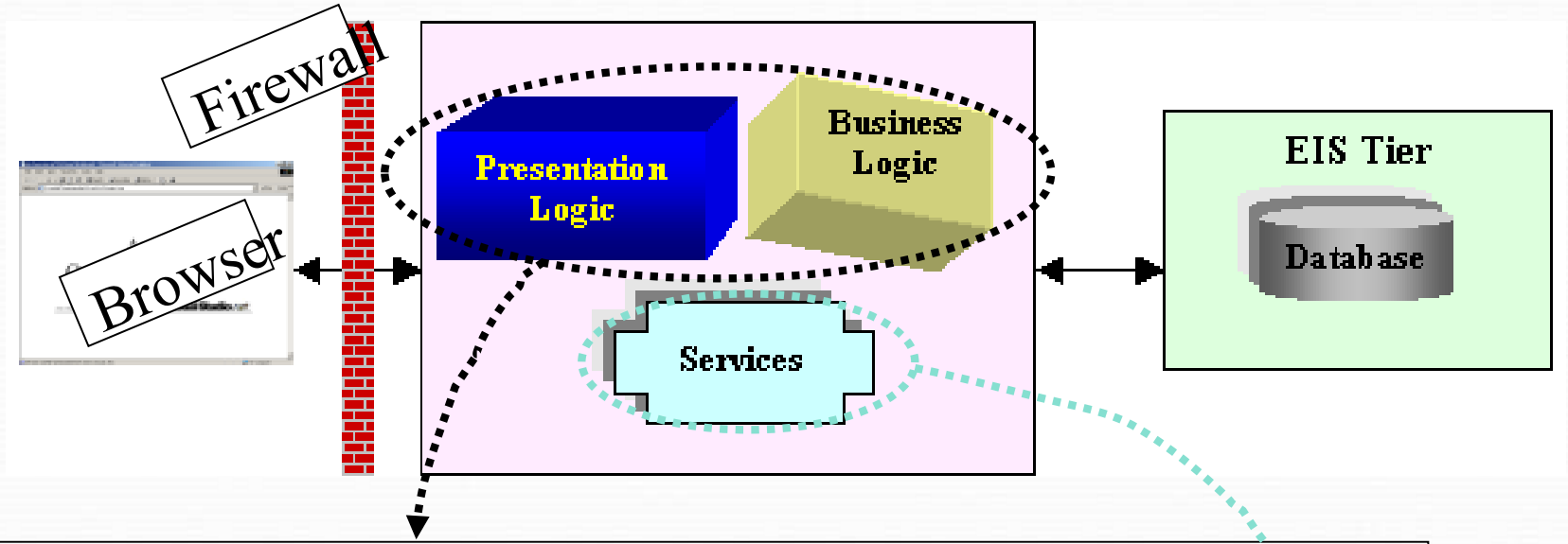
Business logic can be **changed** without impacting clients

Middle tier can be hidden **behind firewall**

Maximum use of available **system resources**

Parts of architecture can be **rewritten** without rewriting whole app

Traditional n-Tier Architecture

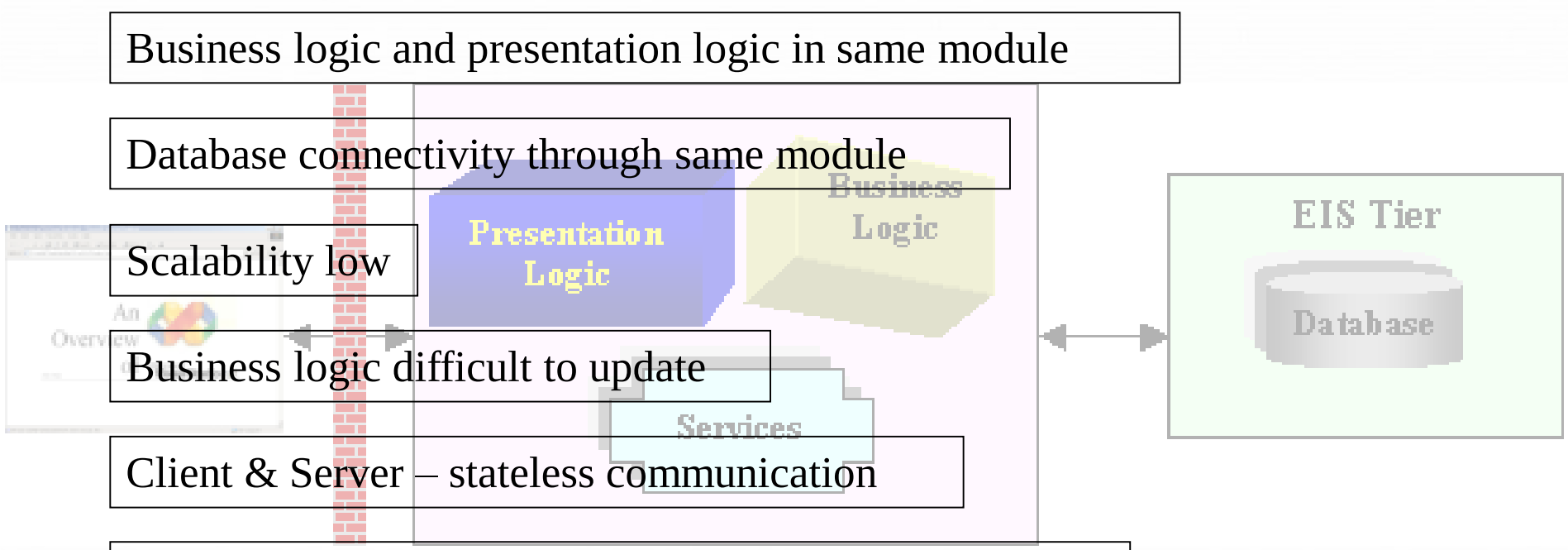


Application Logic= Presentation logic + Business Logic

Infrastructure services provide additional functionalities required by application, such as messaging services and transactional services.

Traditional n-Tier Architecture: The Characteristics

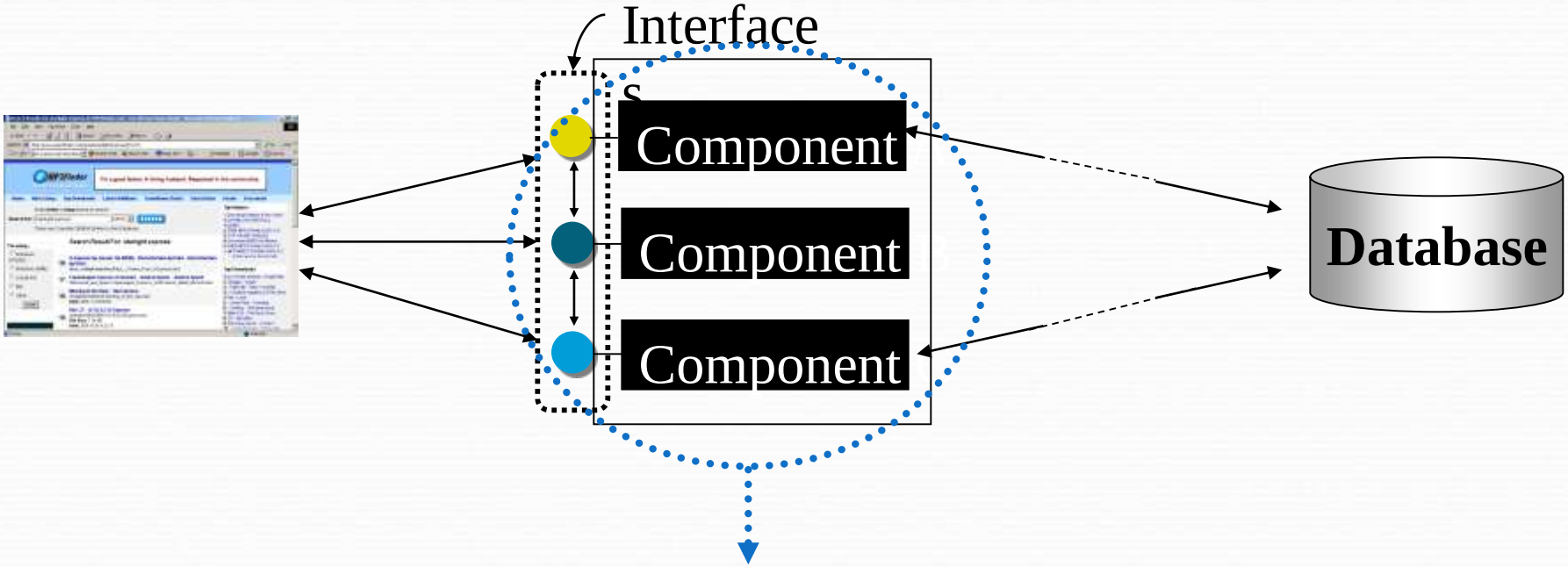
- Business logic and presentation logic in same module
- Database connectivity through same module
- Scalability low
- Business logic difficult to update
- Client & Server – stateless communication
- Business logic unaware of different client identities
- Client has to maintain state



Overcoming the drawbacks - Improving the system

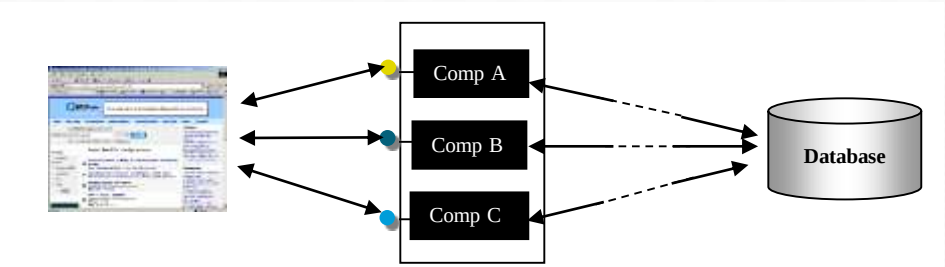
PROBLEM	SOLUTION
Middle Tier contains one App object	Extend the middle tier to create one more layer
For different types of needs- different app objects required	Allow multiple application objects to reside on the server
Different application objects may not be able to communicate with each other	Use interfaces to communicate between application objects

Component n-tier Architecture



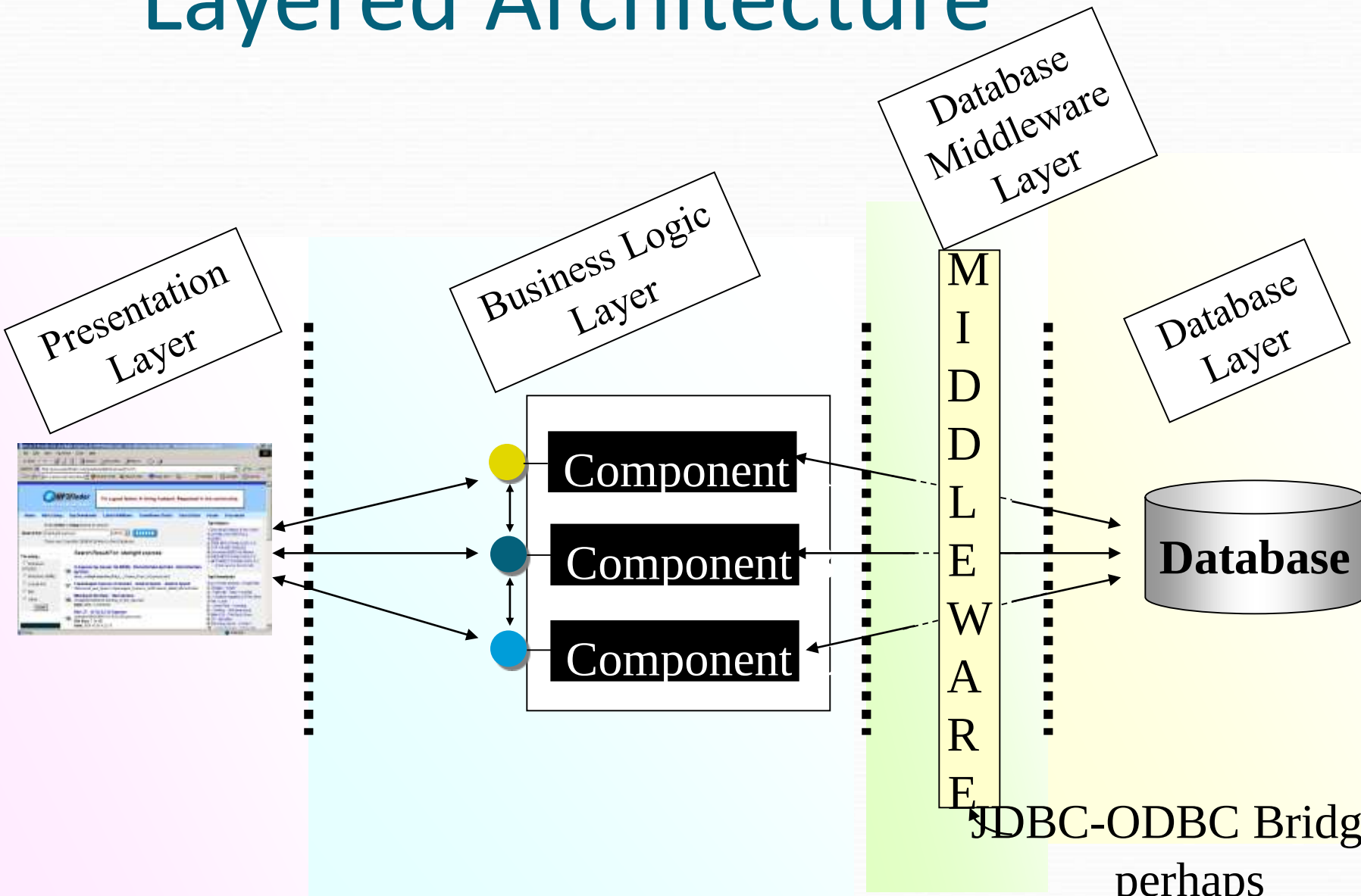
Application object broken into components that can communicate with each other, through interfaces

Component Based n-Tier Systems

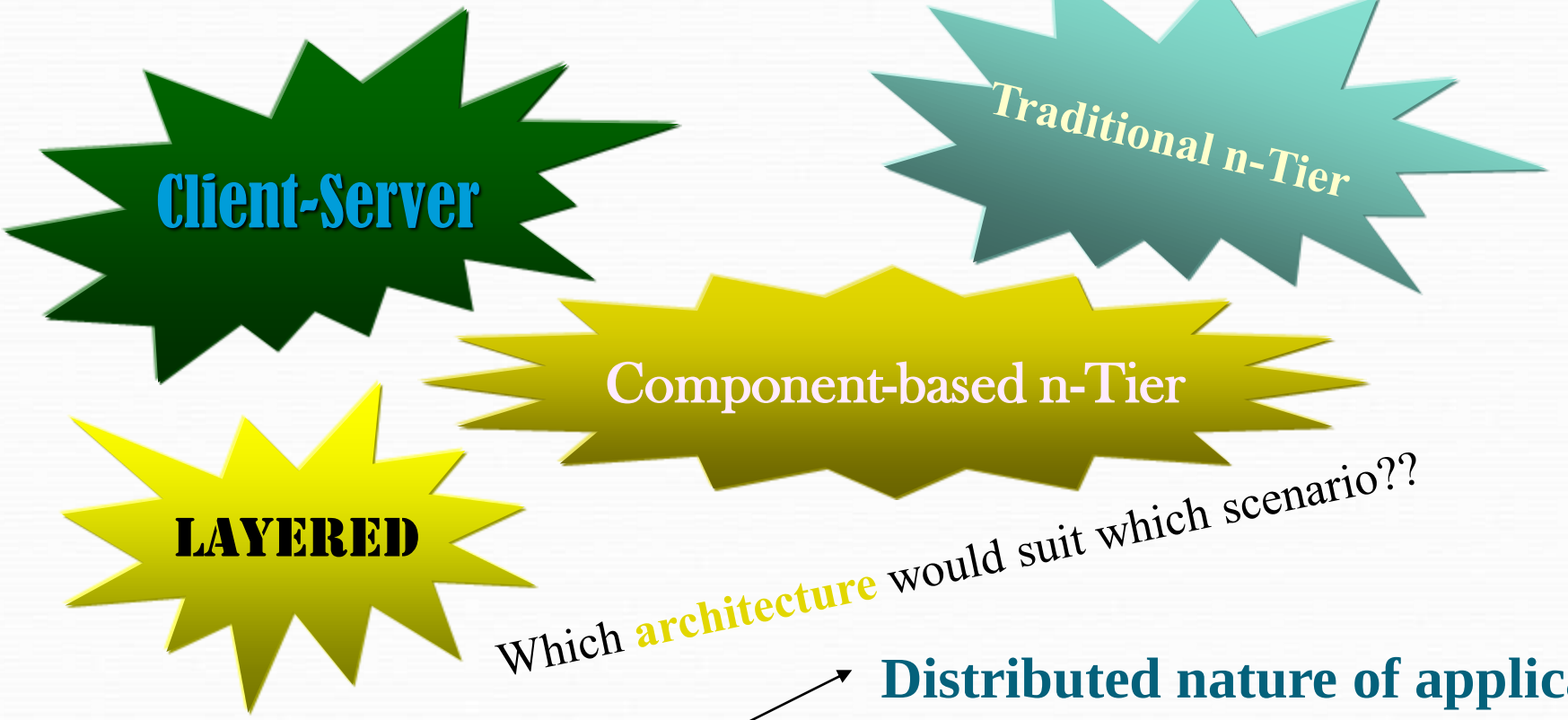


- Component objects maintain identity and encapsulate remote methods
- Components can be designed to maintain session state on server
- Business logic can be modified without affecting other logic

Layered Architecture



Various models of architecture



Which **architecture** would suit which scenario??

Depends on

- **Distributed nature of applicatio**
- **Scalability**
- **Performance**
- **Memory Management**

J2EE Architecture

J2EE is a layered architecture

J2EE framework designed based on…

Using these we can design applications that are…

Flexible

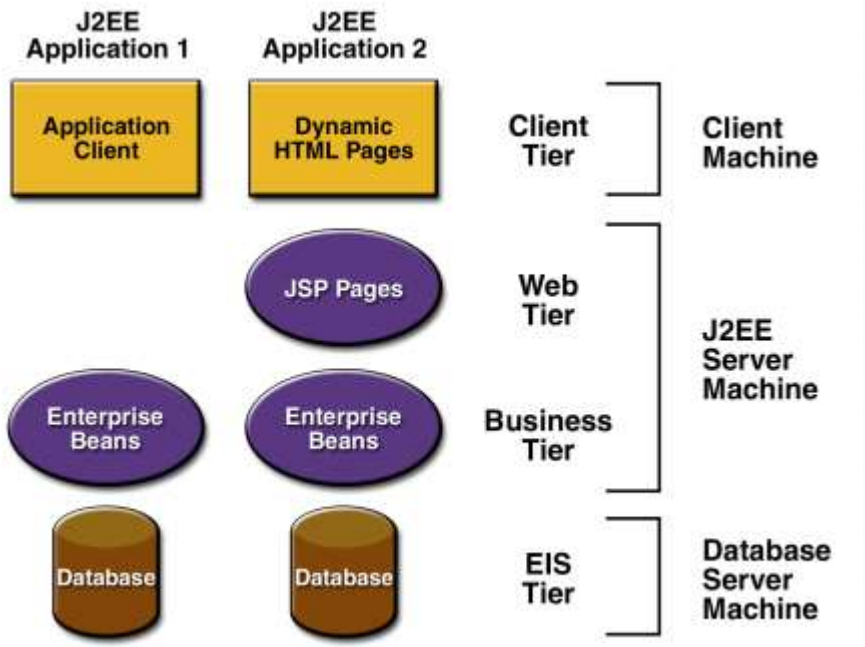
Scalable

Distributed

Component-based

Multi-tier

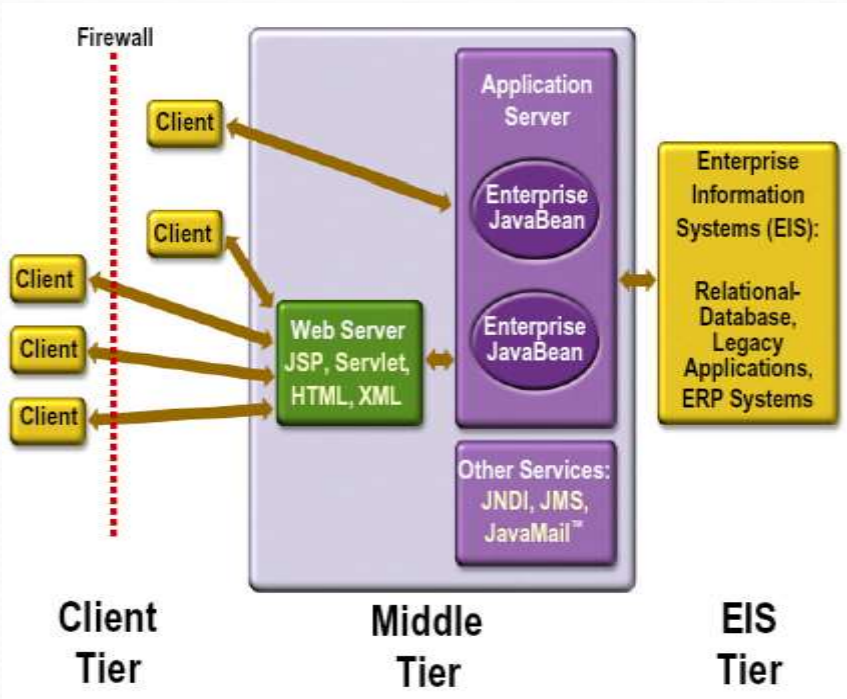
J2EE Architecture



- J2EE multi-tiered applications are generally considered to be three-tiered applications because they are distributed over three different locations
  - client machines
  - the J2EE server machine
  - the database or legacy machines at the back end

J2EE Architecture

- Three-tiered applications that run in this way extend the standard two-tiered client and server model by placing a multithreaded application server between the client application and back-end storage



J2EE goals

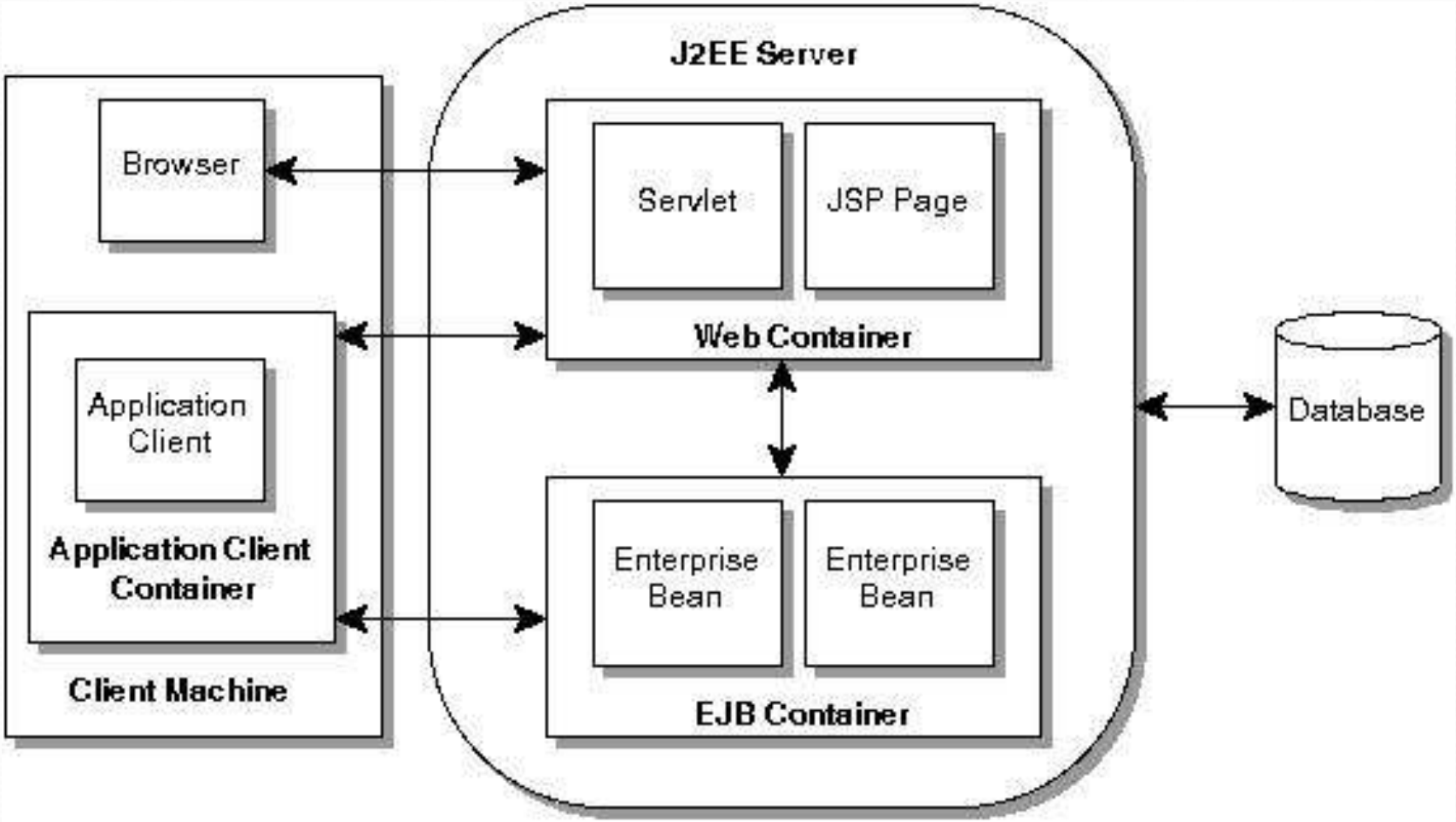
- Robustness
- Scalability
- Simplicity
- Maintainability
- Testability
- Reusability

J2EE Containers

- The application server maintains control and provides services through an interface or framework known as a *container*
- There are five defined container types in the J2EE specification

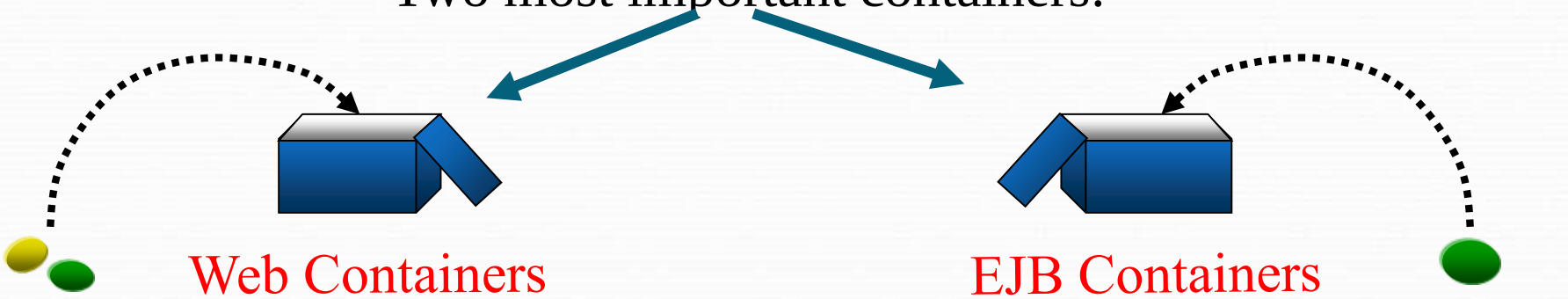
J2EE Containers

- Three of these are server-side containers:
  - The server itself, which provides the J2EE runtime environment and the other two containers
  - An EJB container to manage EJB components
  - A Web container to manage servlets and JSP pages
- The other two container types are client-side:
  - An application container for stand-alone GUIs, console
  - An applet container, meaning a browser, usually with the Java Plug-in


J2EE Server
Browser
Servlet
JSP Page
Web Container
Application Client
Application Client Container
Client Machine
Enterprise Bean
Enterprise Bean
EJB Container
Database

J2EE Container

Two most important containers:



Web Components
(JSP/Servlets)

- Manages threading for components
- Provides necessary interface with web server

- Holds the following components:
  - Entity beans
  - Stateful session beans
  - Stateless session beans
  - Message beans

Containers provide medium for services to communicate with domain layer

J2EE Components

- As said earlier, J2EE applications are made up of components
- A *J2EE component* is a self-contained functional software unit that is assembled into a J2EE application with its related classes and files and that communicates with other components

Components

- Client components run on the client machine, which correlate to the client containers
- Web components -servlets and JSP pages
- EJB Components

Packaging Applications and Components

- Under J2EE, applications and components reside in Java Archive (JAR) files
- These JARs are named with different extensions to denote their purpose, and the terminology is important

Various File types

- Enterprise Archive (EAR) files represent the application, and contain all other server-side component archives that comprise the application
- Client interface files and EJB components reside in JAR files
- Web components reside in Web Archive (WAR) files

Deployment Descriptors

- Deployment descriptors are included in the JARs, along with component-related resources
- Deployment descriptors are XML documents that describe configuration and other deployment settings (remember that the J2EE application server controls many functional aspects of the services it provides)
- The statements in the deployment descriptor are declarative instructions to the J2EE container; for example, transactional settings are defined in the deployment descriptor and implemented by the J2EE container

EJB Components

- EJB components are server-side, modular, and reusable, comprising specific units of functionality
- They are similar to the Java classes we create every day, but are subject to special restrictions and must provide specific interfaces for container and client use and access
- We should consider using EJB components for applications that require scalability, transactional processing, or availability to multiple client types

EJB Components- Major Types

- **Session beans**
  - These may be either *stateful* or *stateless* and are primarily used to encapsulate business logic, carry out tasks on behalf of a client, and act as controllers or managers for other beans
- **Entity beans**
  - Entity beans represent persistent objects or business concepts that exist beyond a specific application's lifetime; they are typically stored in a relational database