Java 2 Enterprise Edition (J2EE)

March 07st, 2019

Session Objectives

- Understanding the value propositions of J2EE
- Getting a big picture of J2EE architecture and platform
- Getting high-level exposure of APIs and Technologies that constitute J2EE
  - You don't have to understand all the details
- Understanding why J2EE can be used for as a platform for development and deployment of web services

Agenda

- Introduction to J2EE
- J2EE Framework
- Support to J2EE of big software vendors
- Software Architectures
- Features and Concepts in J2EE
- Sample J2EE Architectures

Enterprise Computing

Challenges
- Portability
- Diverse Environments
- Time-to-market
- Core Competence
- Assembly
- Integration

Key Technologies
- J2SE™
- J2EE™
  - JMS
  - Servlet
  - JSP
  - Connector
- XML
  - Data Binding
  - XSLT

Products
- App Servers
- Web Servers
- Components
- Databases
- Object to DB tools

Legacy Systems
- Databases
- TP Monitors
- EIS Systems

J2SE: Java Second Standard Edition
JMS: Java Messages Service
TP Monitors: Transaction Processing Monitors
EIS: Execute Information System

What Is J2EE?

- Say simply, J2EE is:
  - a suite of *specifications* for application programming interfaces
  - a distributed computing architecture
  - definitions for packaging of distributable components for deployment.
- It's a collection of standardized **components**, **containers**, and **services** for creating and deploying distributed applications within a well-defined distributed computing architecture.

What Is J2EE?

- Open and standard based platform for
  - developing, deploying and managing
  - n-tier, Web-enabled, server-centric, and component-based enterprise applications

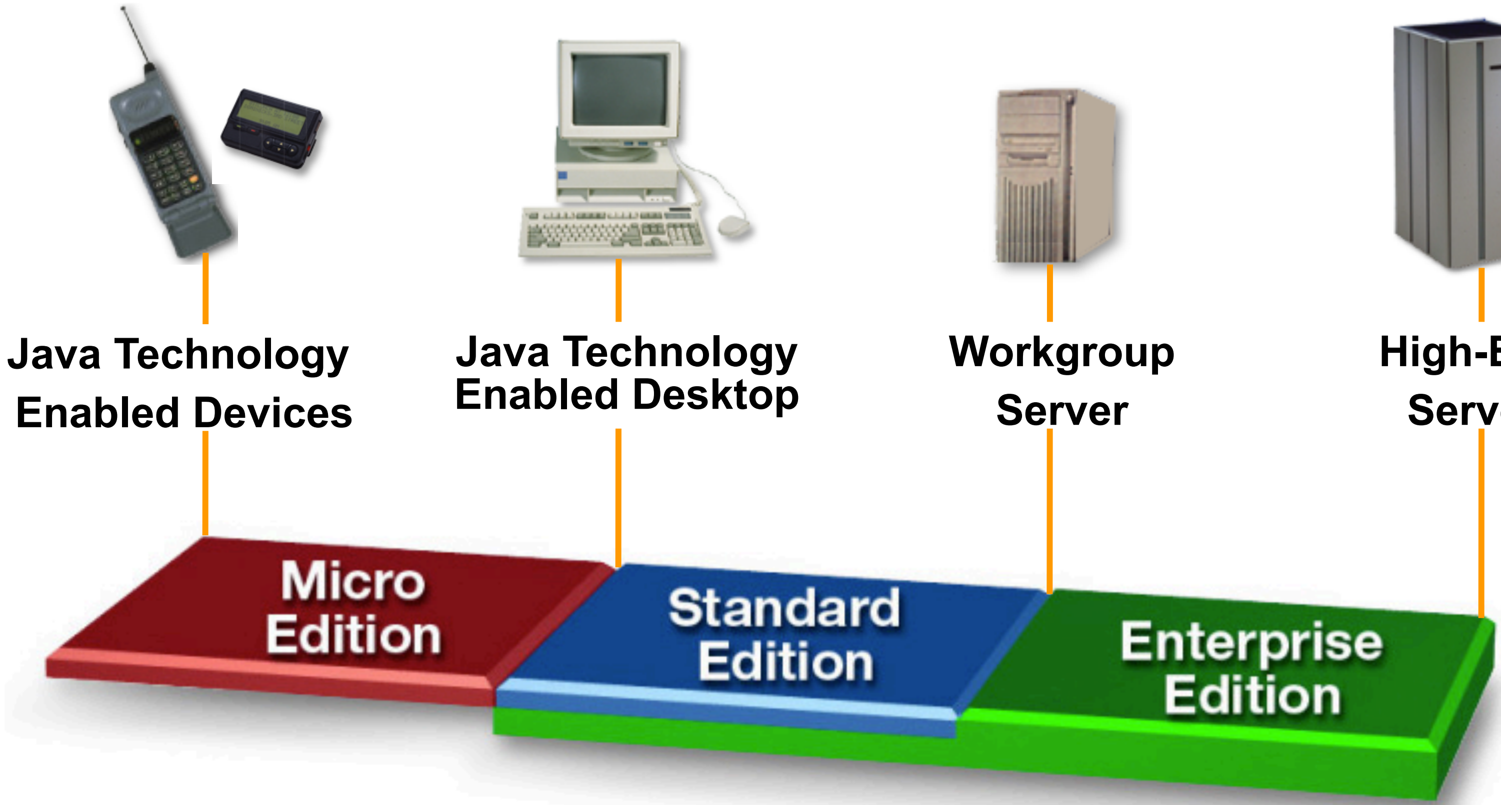
In short:

J2EE is an open, standard-based, development and deployment platform for building n-tier, web-based and server-centric, and component-based enterprise applications.

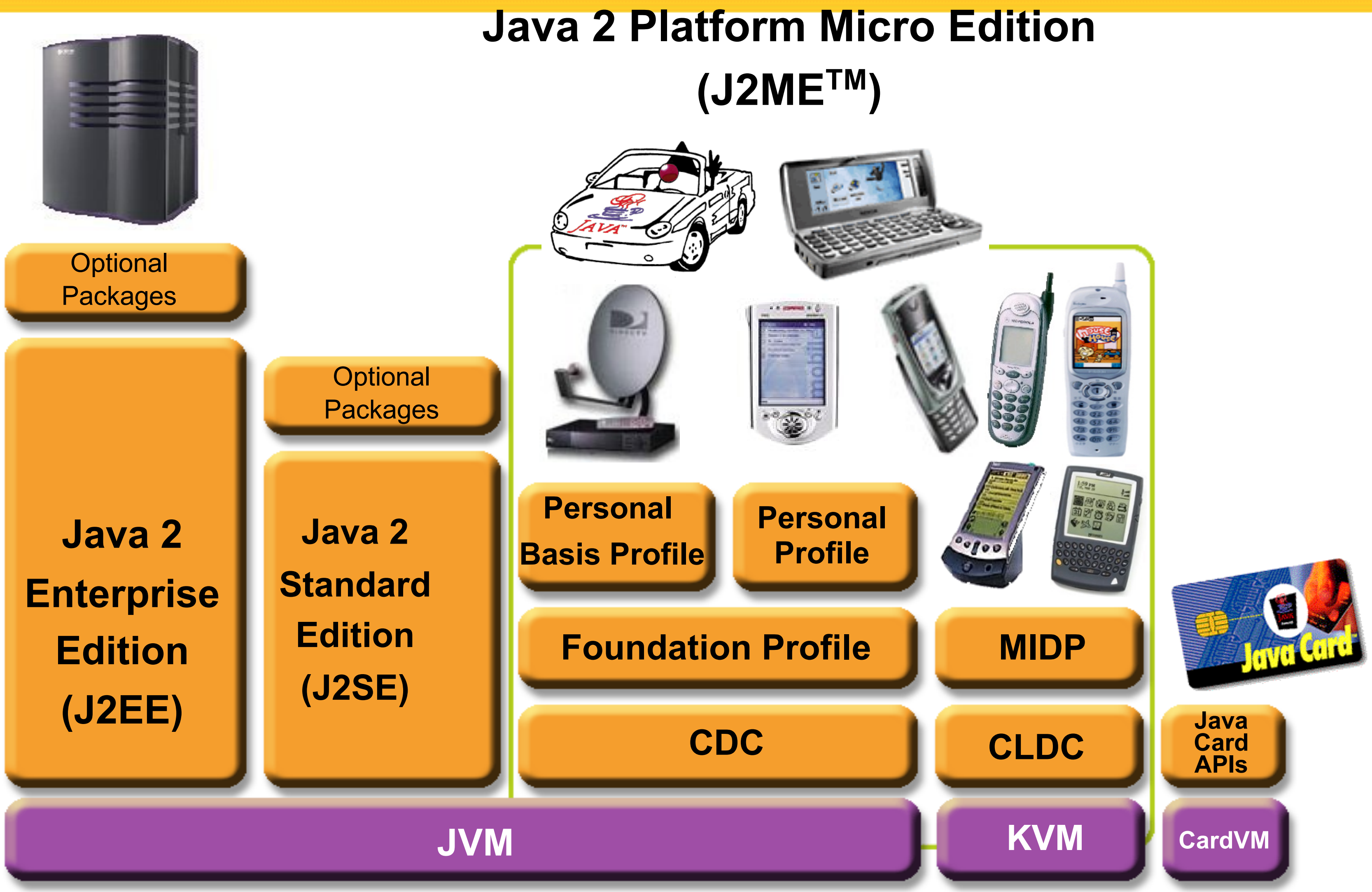
What does J2EE comprise?

- Java Servlets
- JavaServer Pages (JSP)
- Enterprise JavaBeans (EJB)
- Java Message Service (JMS)
- Java Naming and Directory Interface (JNDI)
- Java Database Connectivity (JDBC)
- JavaMail
- Java Transaction Service (JTS)
- Java Transaction API (JTA)
- J2EE Connector Architecture (J2EE-CA, or JCA)

The Java™ Platform



The Java™ Platform



What Makes Up J2EE?

- API and Technology specifications
- Development and Deployment Platform
- Standard and production-quality implementation
- Compatibility Test Suite (CTS)
- J2EE brand
- J2EE Blueprints
- Sample codes

Open and Standard Solution

- Use "component and container" model in which container provides system services in a well-defined and as industry standard
- J2EE is that standard that also provides portability of code because it is based on Java technology and standard-based Java programming APIs

When using J2EE?

- J2EE targets at **large-scale** business systems
- The software in the J2EE framework needs to be partitioned into functional pieces and deployed on the appropriate hardware platforms to provide the necessary computing power
- J2EE provides:
  - a collection of standardized components that facilitate software deployment
  - standard interfaces that define how the various software modules interconnect
  - standard services that define how the different software modules communicate.

Relate to J2SE

- J2SE (Java 2 Standard Edition) is the **core** upon which J2EE is based
- Use J2SE components and APIs in conjunction with the J2EE components and APIs to build your applications

Why using J2EE?

J2EE:

- Define a number of essential services to develop enterprise-class applications
- Provide infrastructure required to write enterprise-class applications: there are a bunch of different system-level capabilities to write distributed applications that are scalable, robust, secure, and maintainable
- Define a set of containers, connectors, and components that can run on any number of J2EE-compliant implementations

Platform Value to Developers

- Can use ***any J2EE implementation*** for development and deployment
  – Use production-quality standard implementation which is free for development/deployment
  – Use high-end commercial J2EE products for scalability and fault-tolerance
- Vast amount of J2EE ***community resources***
  – Many J2EE related books, articles, tutorials, quality code you can use, best practice guidelines, design patterns etc.
- Can use off-the-shelf 3rd-party business components

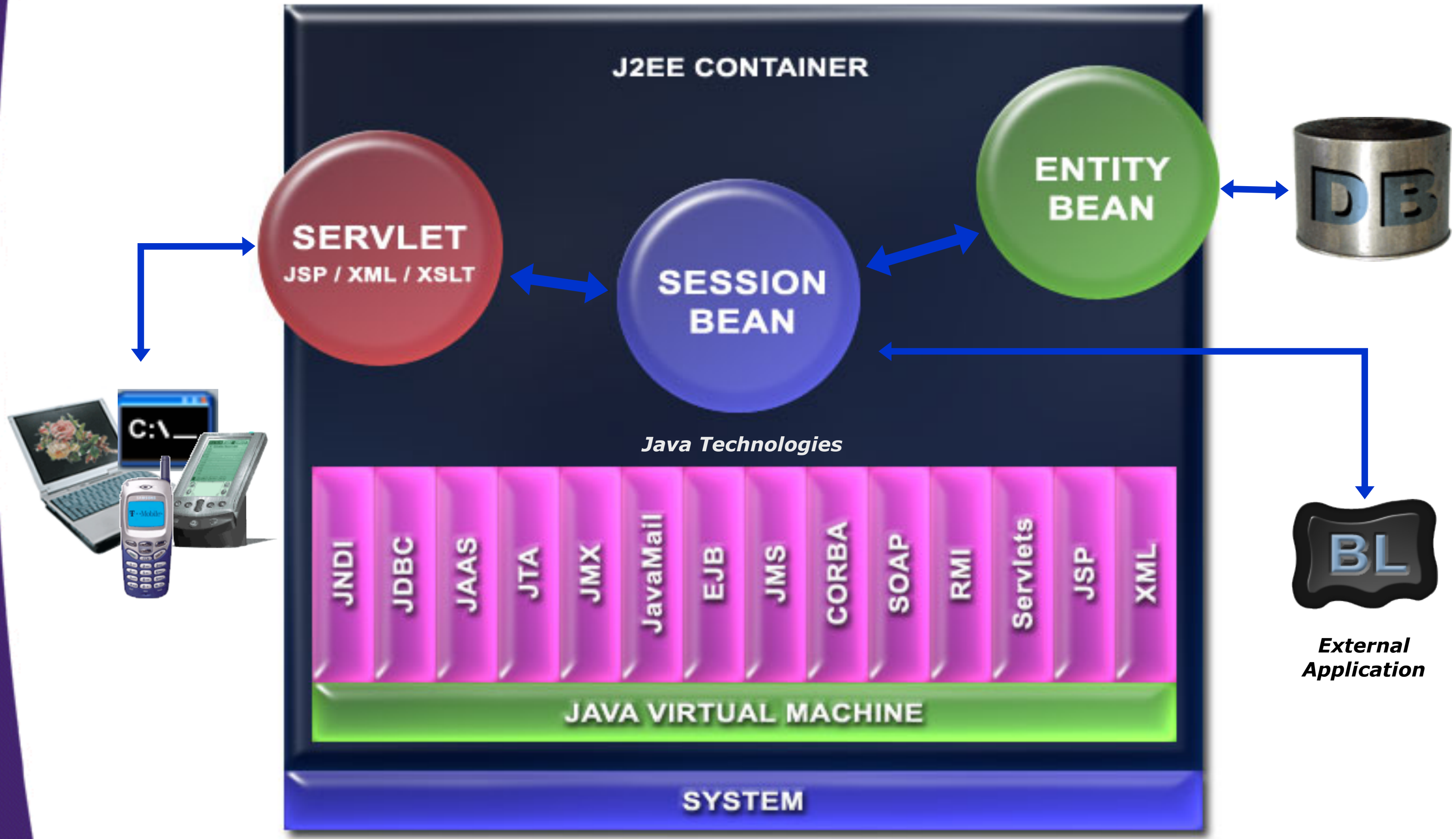
Platform Value to Vendors

- Vendors work together on specifications and then compete in implementations
  - In the areas of Scalability, Performance, Reliability, Availability, Management and development tools, and so on
- Freedom to innovate while maintaining the portability of applications
- ***Do not have create/maintain their own proprietary APIs***

Platform Value to Business Customers

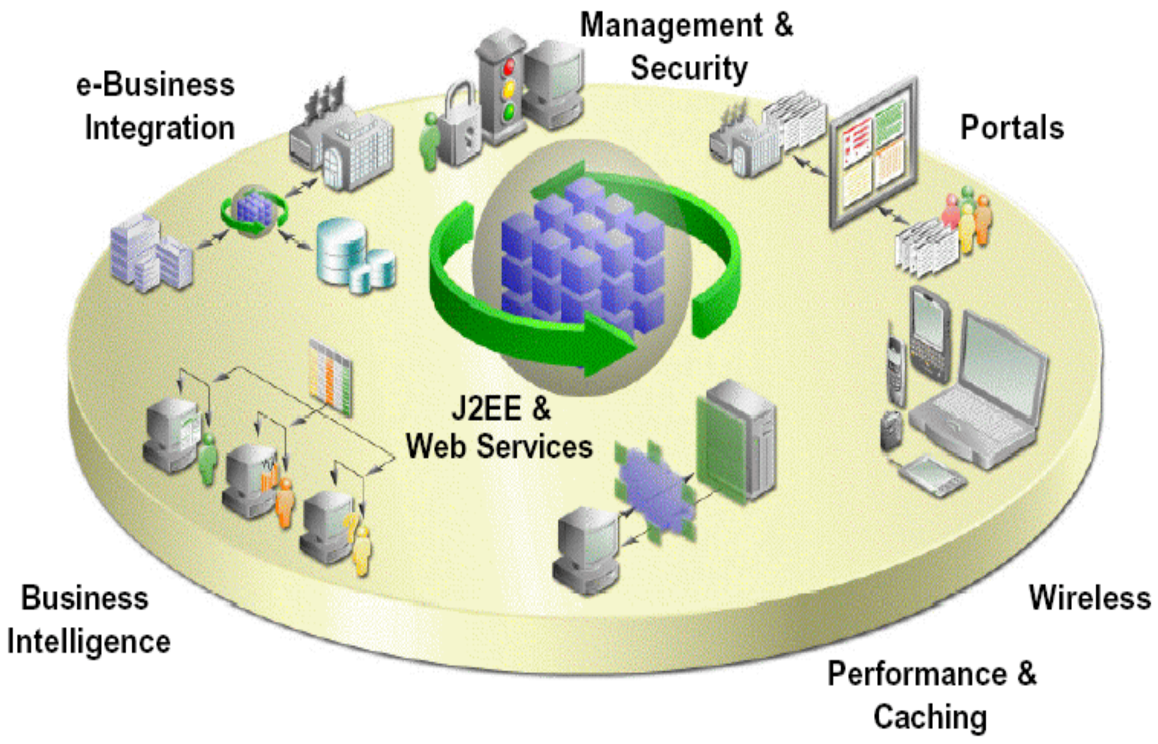
- ***Application portability***
- Many implementation choices are possible based on various requirements
  - Price (free to high-end), scalability (single CPU to clustered model), reliability, performance, tools, and more
  - Best of breed of applications and platforms
- Large developer pool

J2EE Framework



ORACLE

- Oracle 9*i*AS Internet Application Server Enterprise Edition used by SCT for Banner
- 100% compliant J2EE server

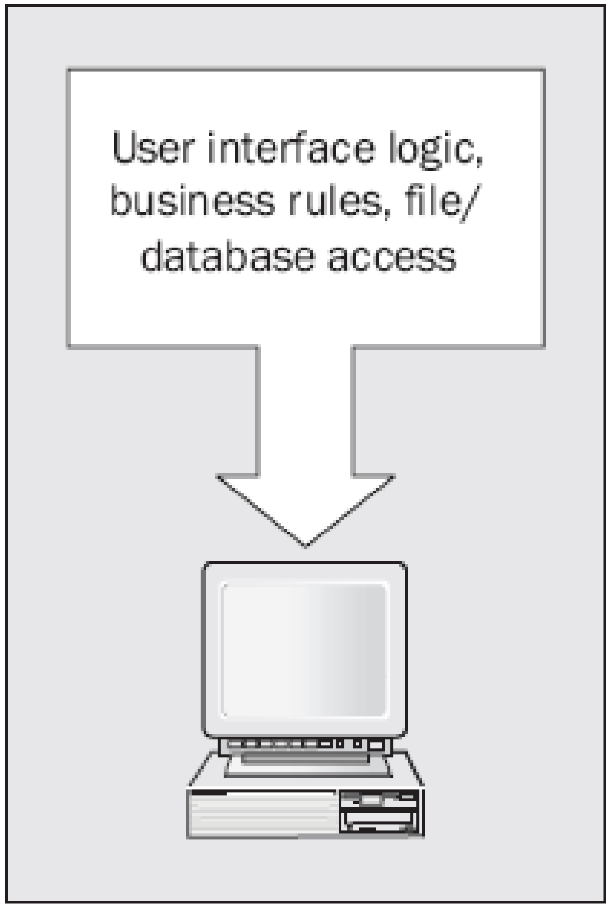


Oracle9i Application Server
http://www.oracle.com/ip/deploy/ias

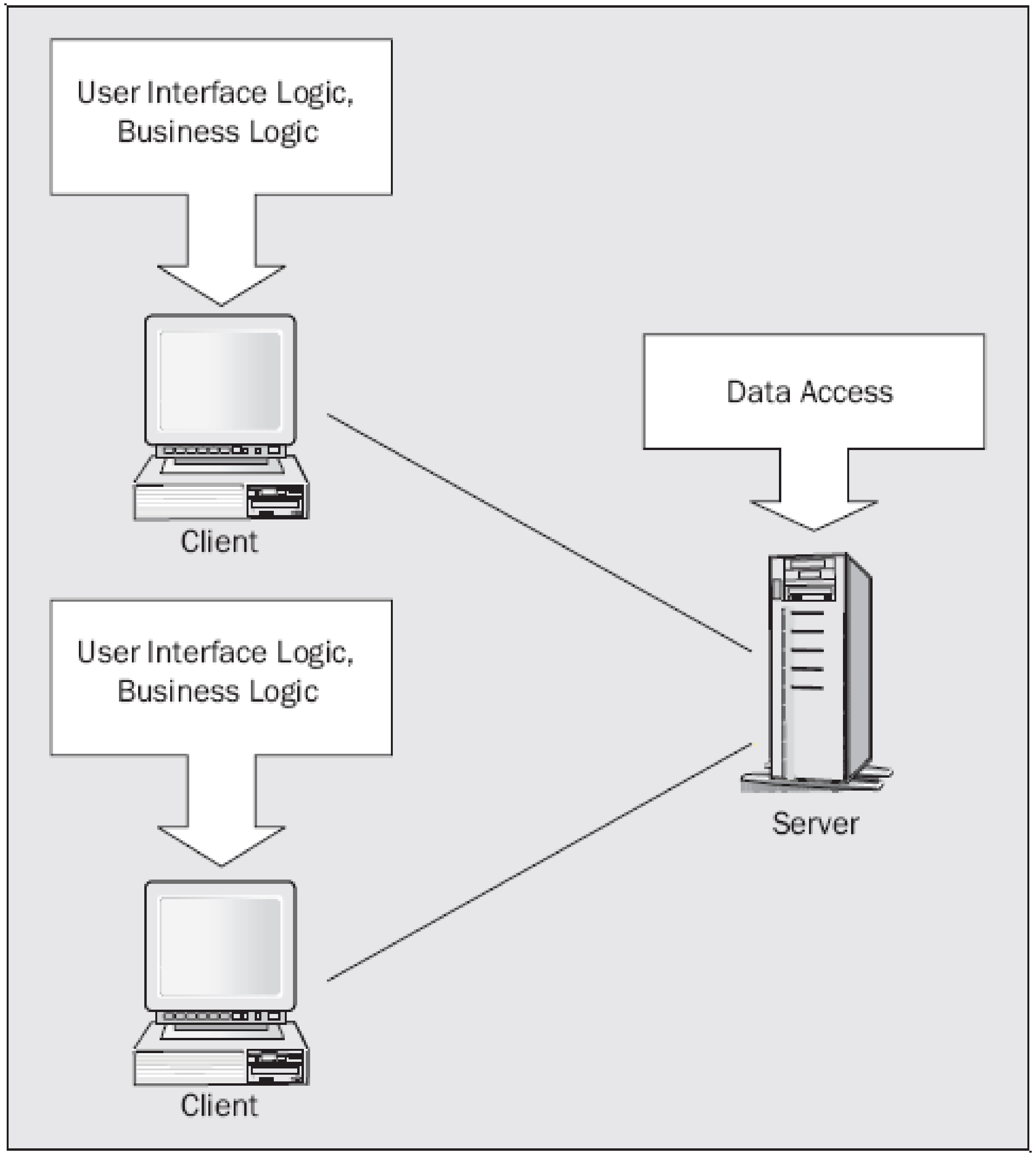
IBM

- "WebSphere continues the evolution to a single Web services-enabled, Java™ 2 Enterprise Edition (J2EE) application server and development environment that addresses the essential elements needed for an on demand operating environment."
  - *http://www-3.ibm.com/software/info1/websphere*
- IBM & Globus Project developing grid computing with JBoss and IBM WebSphere
  - *http://www-1.ibm.com/grid/grid_strategy.shtml*
  - *http://www.javaworld.com/javaworld/jw-09-2002/jw-0906-grid.html*

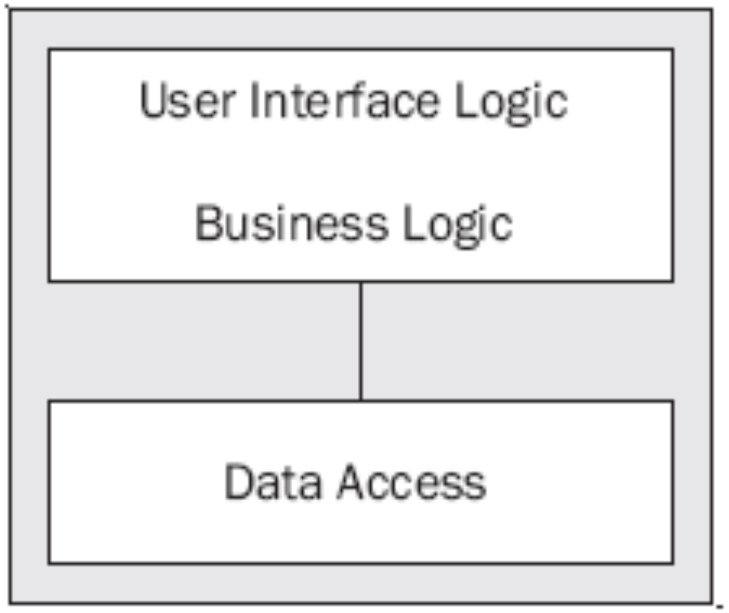
Single Tier Applications



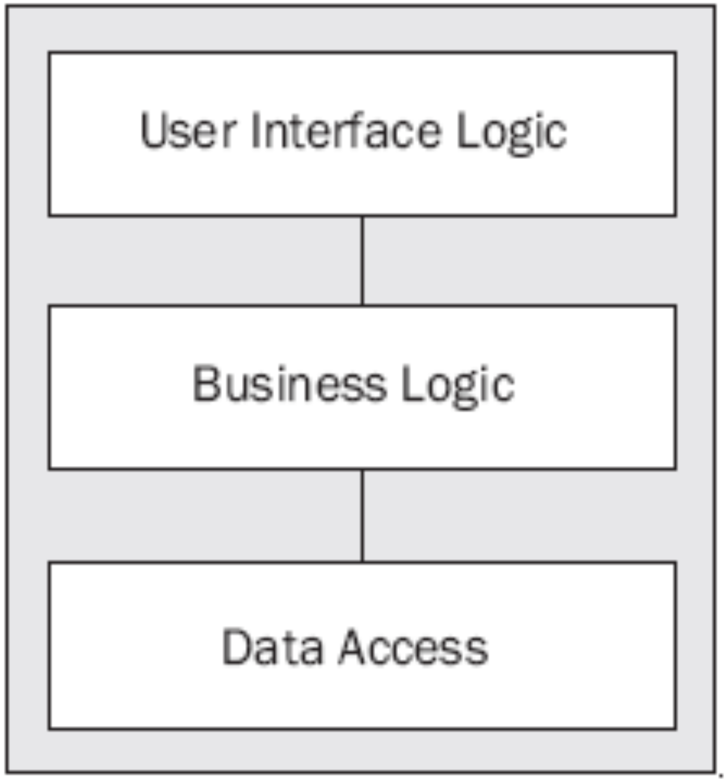
Client-Server Applications


User Interface Logic, Business Logic
Client
User Interface Logic, Business Logic
Client
Data Access
Server

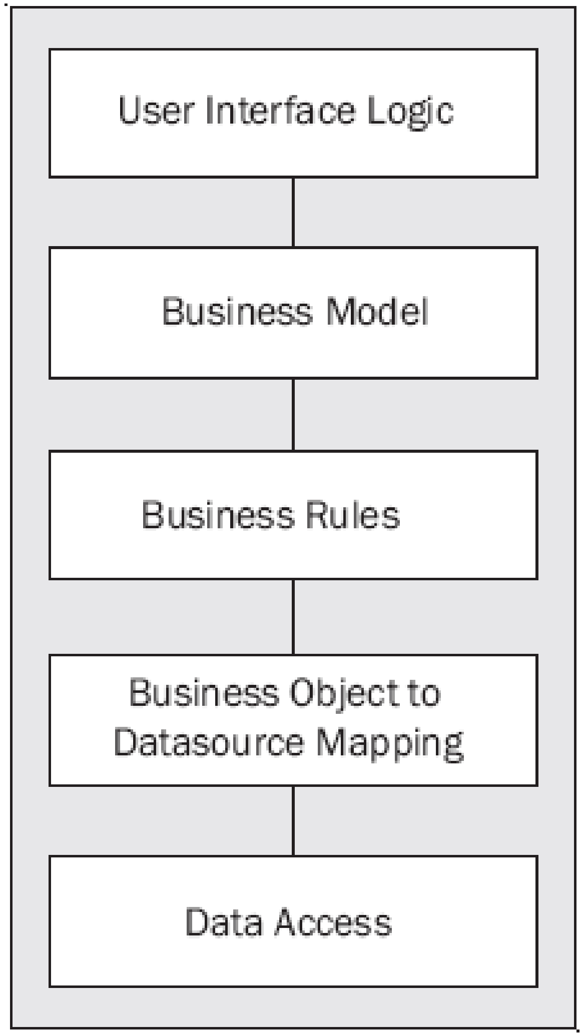
Two-Tier Architecture



Three-Tier Architecture



Multi-Tier Architecture



Vendor Independence

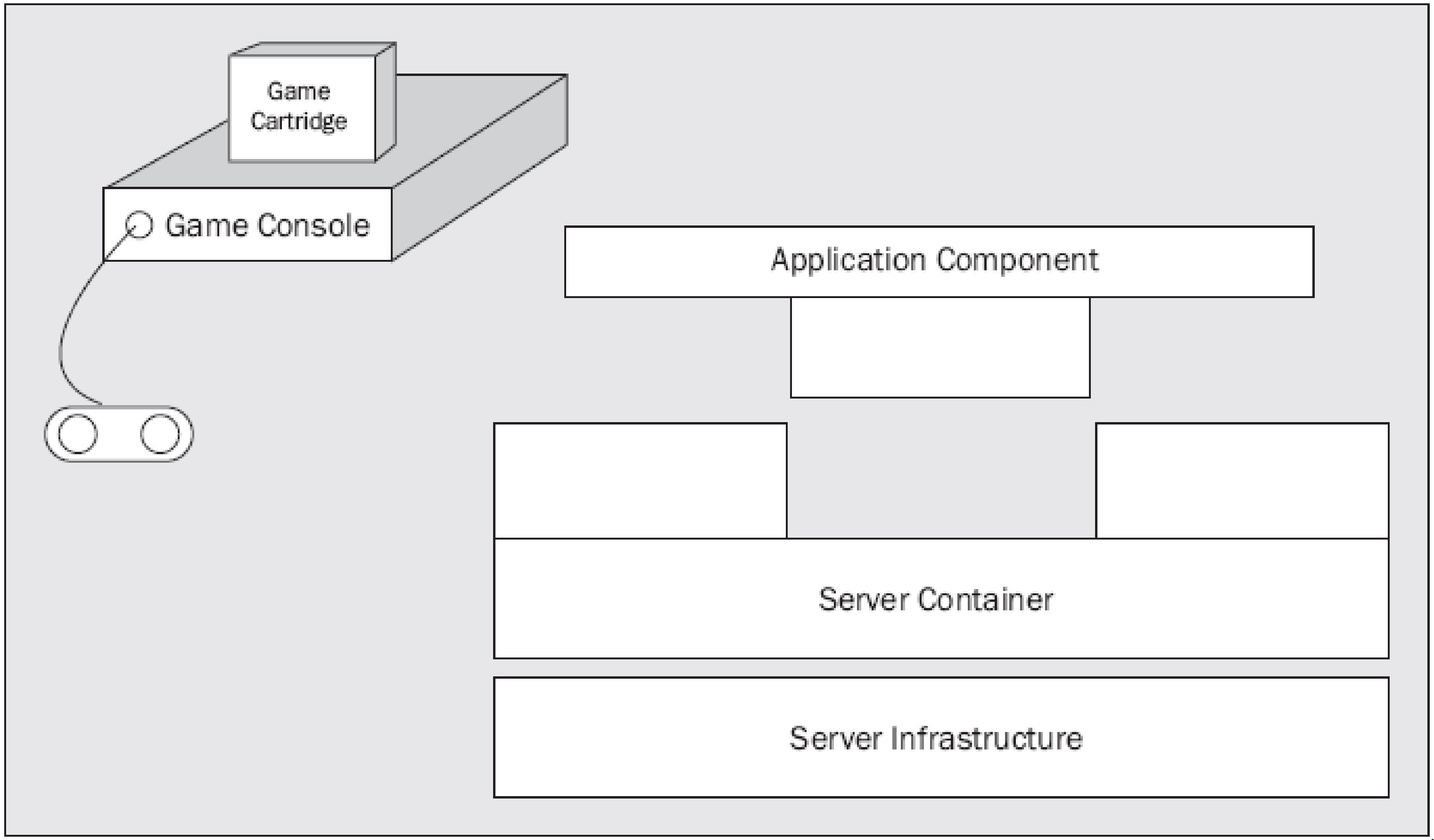
- Java (including J2EE) is designed to run on all platforms (platform-independence)
- The architects of J2EE has an open specification that can be implemented by vendors

Scalability

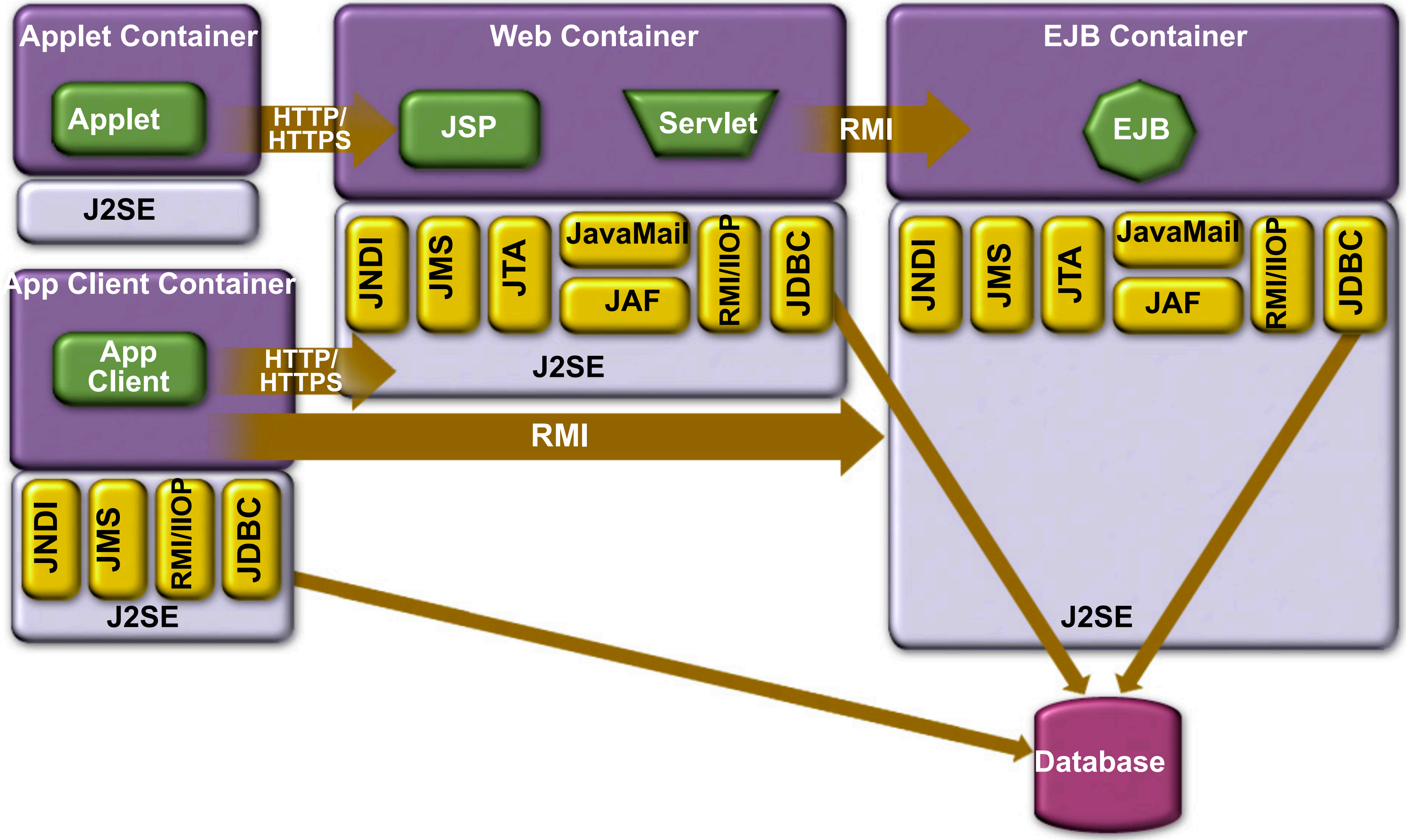
- Changes in requirements → changes have to be made in software
- The J2EE architecture provides much flexibility to accommodate changes as the requirements for throughput, performance, and capacity change
- J2EE also supports clustering, connection pooling, and failover

Containers

- A central theme in the J2EE architecture



J2EE Containers & Components



Containers Handle

- Concurrency
- Security
- Availability
- Scalability
- Persistence
- Transaction
- Life-cycle management
- Management

Components Handle

- Presentation
- Business Logic

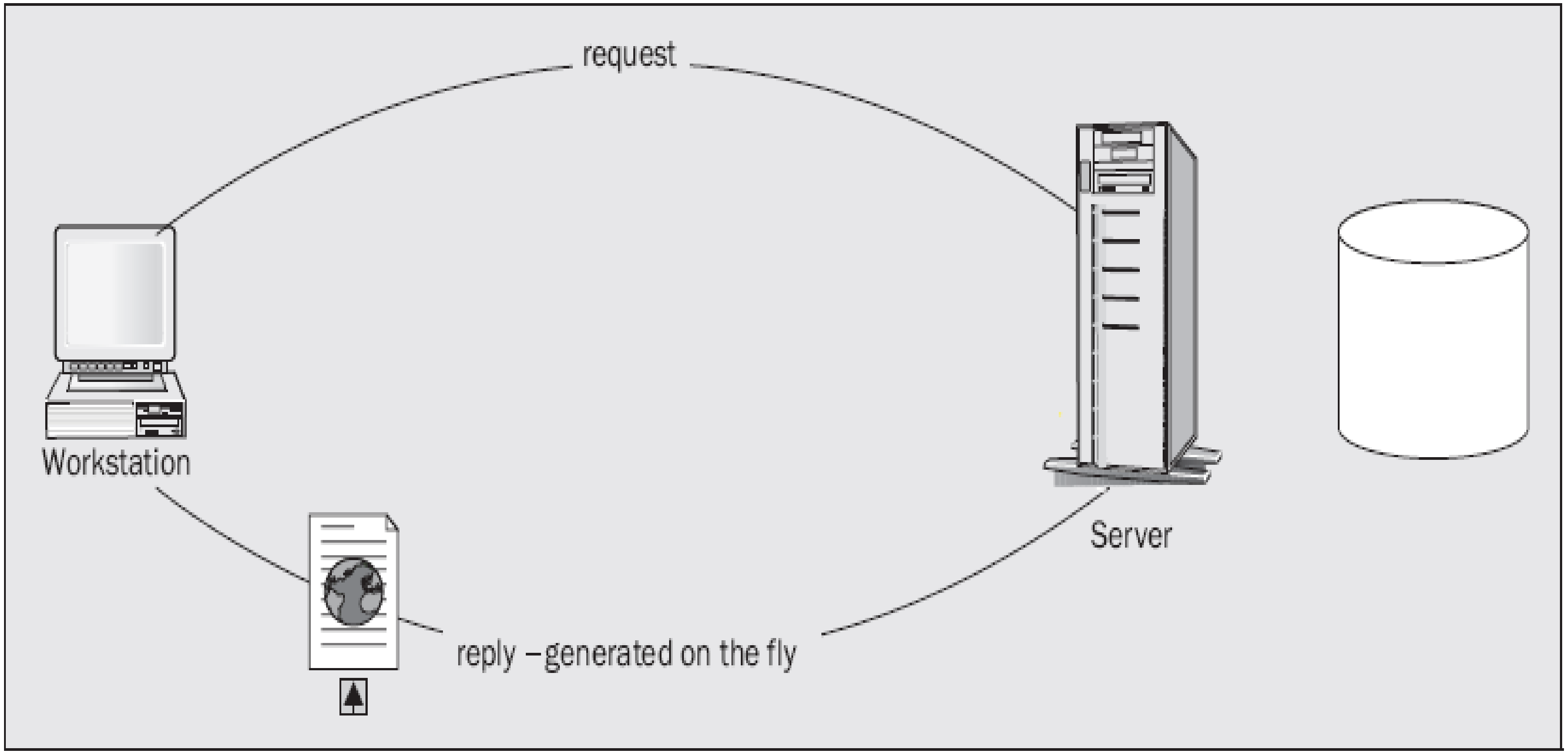
Containers & Components

- Containers do their work invisibly
  - No complicated APIs
  - They control by interposition
- Containers implement J2EE
  - Look the same to components
  - Vendors making the containers have great freedom to innovate

What is a Servlet?

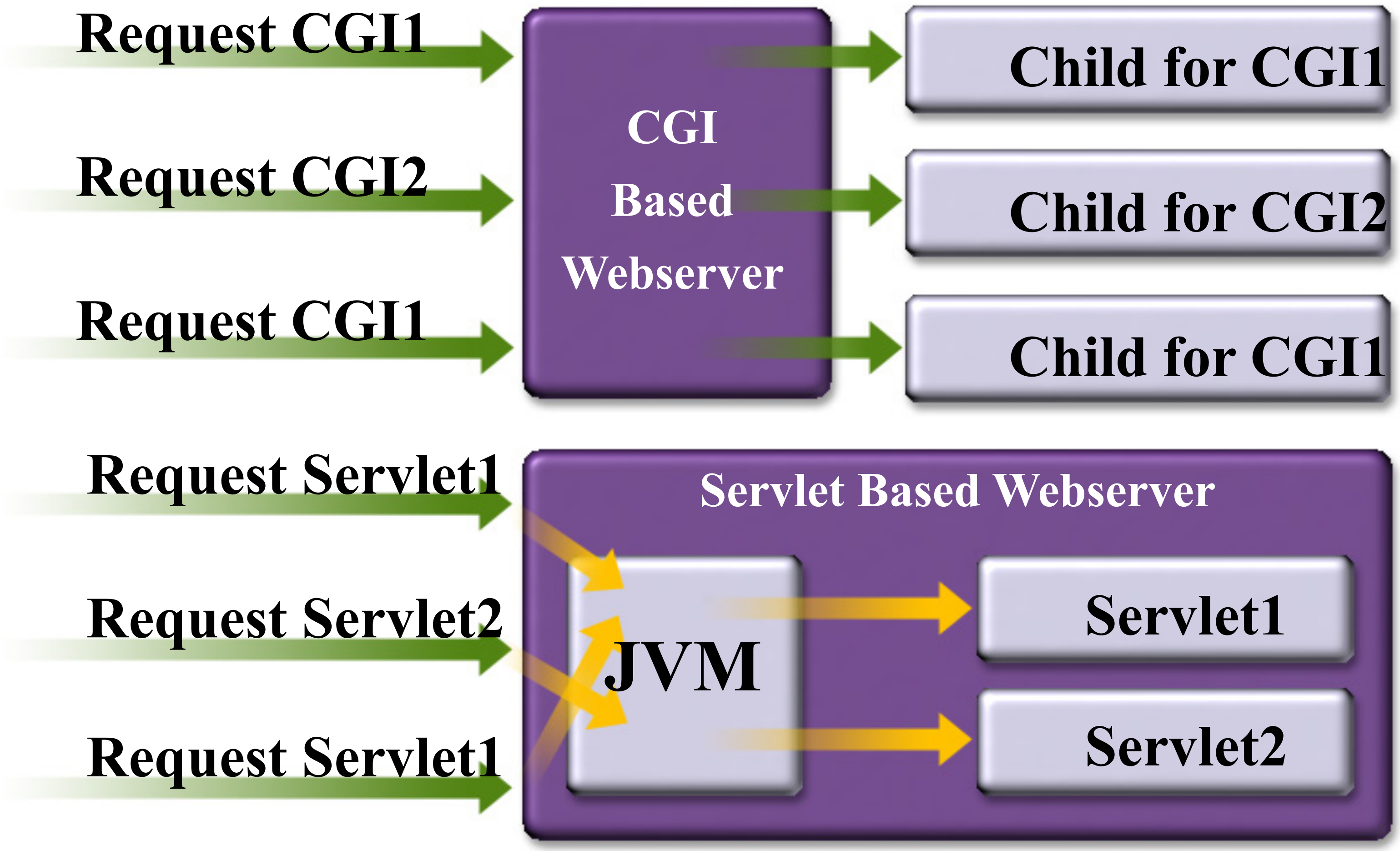
- Java™ objects which extend the functionality of a HTTP server by providing the capability of dynamic contents generation
- Better alternative to CGI, NSAPI, ISAPI, etc.
  - Efficient
  - Platform and server independent
  - Session management
  - Java-based

Java Servlets

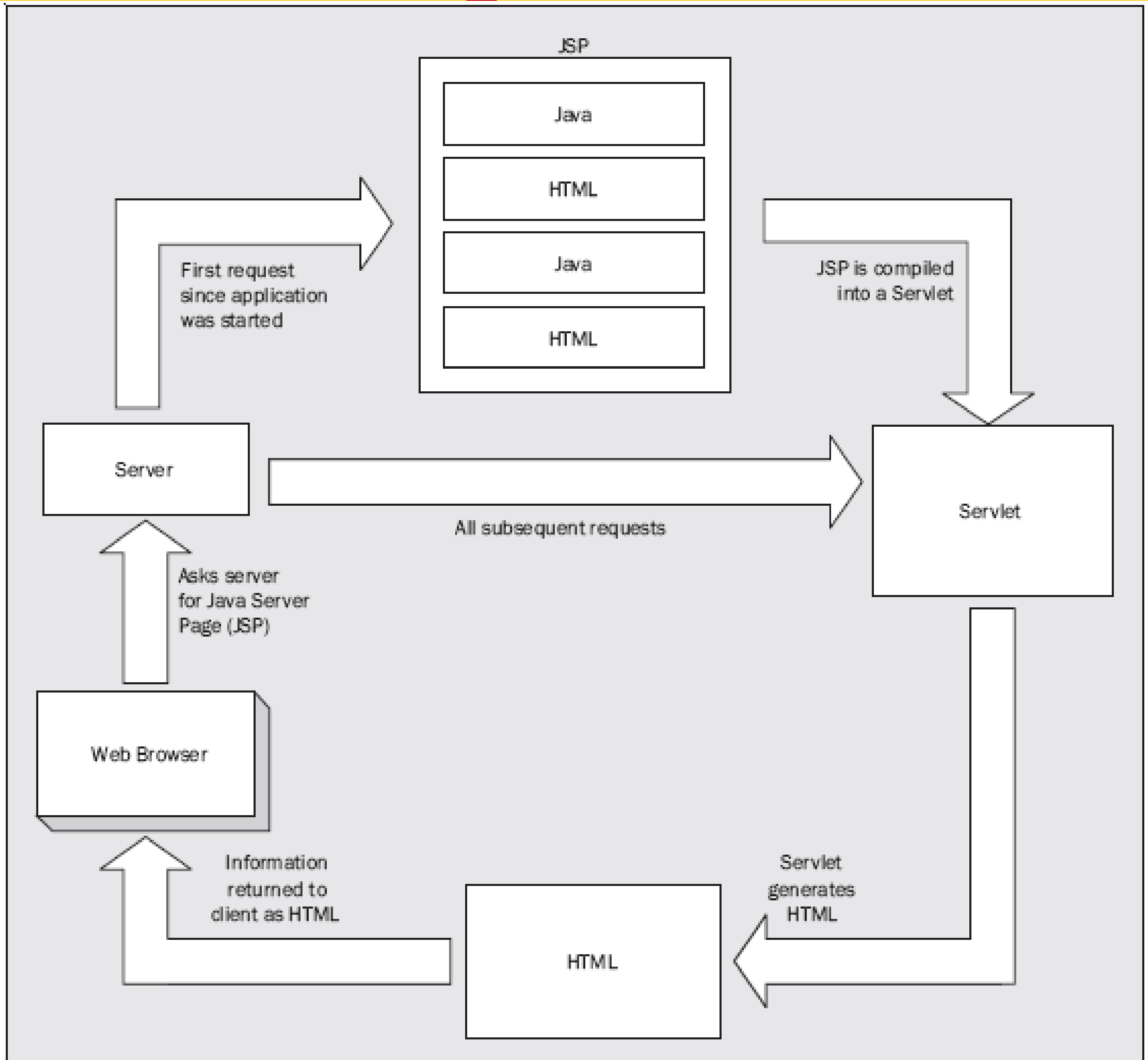


- Provide for dynamically generated content

Servlet vs. CGI



JavaServer Pages


JSP
Java
HTML
Java
HTML
First request since application was started
JSP is compiled into a Servlet
Server
All subsequent requests
Servlet
Asks server for Java Server Page (JSP)
Web Browser
Information returned to client as HTML
Servlet generates HTML
HTML

Enterprise JavaBean (EJB)

- Developed based on Remote Method Invocation (RMI)
- EJBs are Java components that implement business logic. This allows the business logic of an application kept separate from the front-end applications that use that business logic
- The J2EE architecture includes a server that is a container for EJBs
- 3 types:
  - Session bean: maintain the state of sessions
  - Entity bean: represent business objects
  - Message bean: a component model for services that listen to Message Service messages

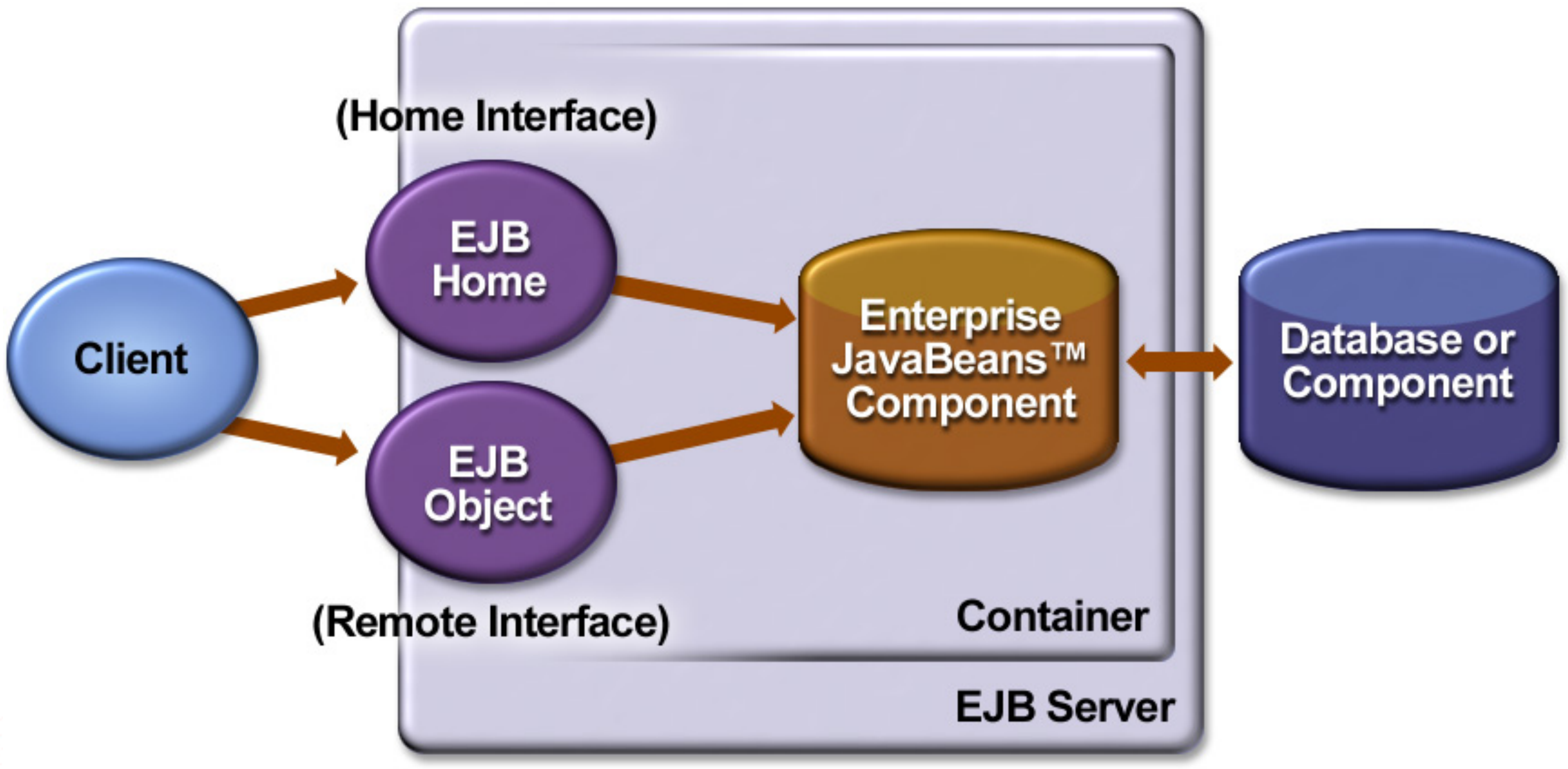
What is EJB Technology?

- A server-side component technology
- Easy development and deployment of Java technology-based application that are:
  - Transactional, distributed, multi-tier, portable, scalable, secure, …

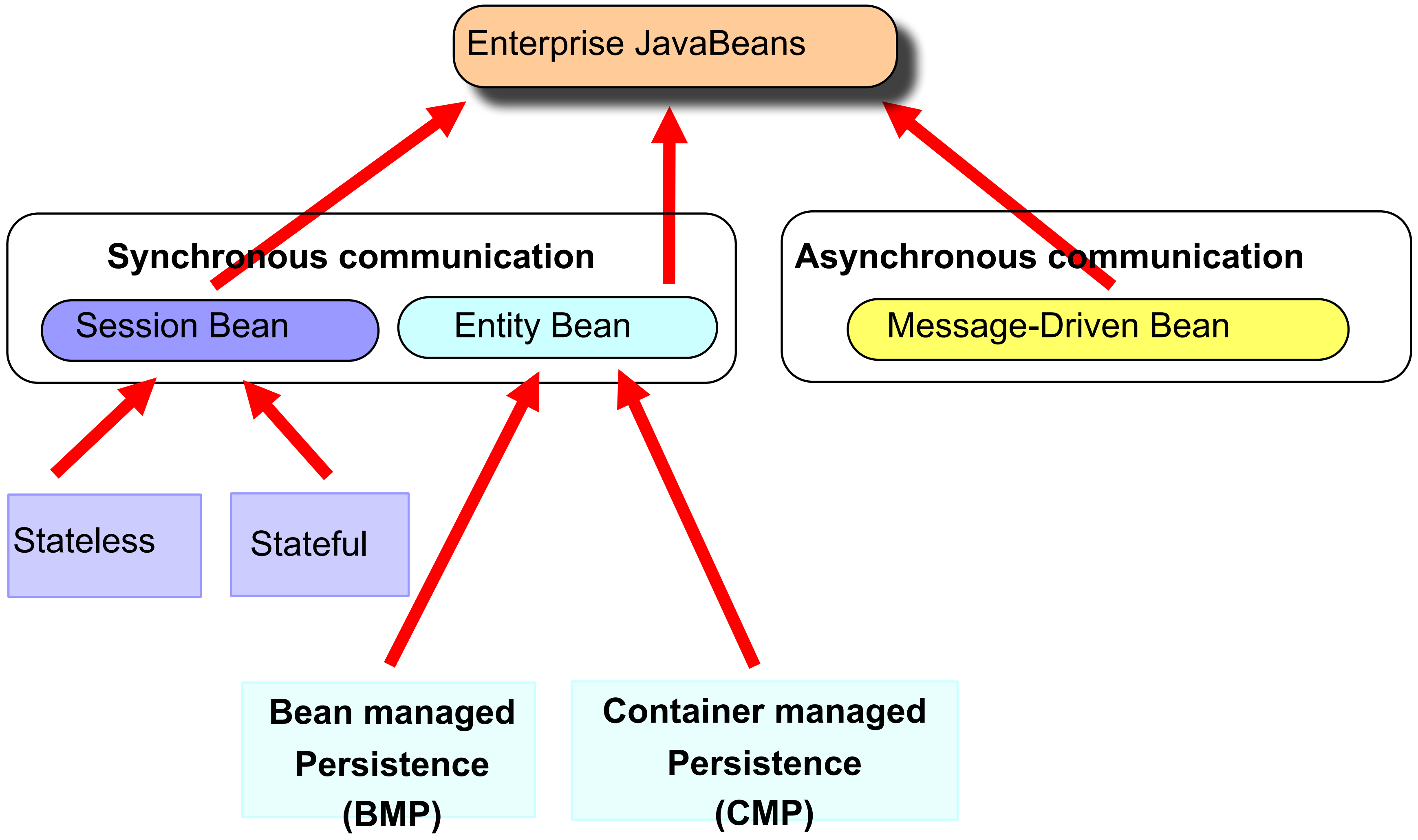
Why EJB Technology?

- Leverages the benefits of component-model on the server side
- Separates business logic from system code
  - Container provides system services
- Provides framework for portable components
  - Over different J2EE-compliant servers
  - Over different operational environments
- Enables deployment-time configuration
  - Deployment descriptor

EJB Architecture



Enterprise JavaBeans


Enterprise JavaBeans
Synchronous communication
Session Bean
Entity Bean
Asynchronous communication
Message-Driven Bean
Stateless
Stateful
Bean managed Persistence (BMP)
Container managed Persistence (CMP)

XML Support

- Extensible Markup Language (XML) is a significant cornerstone for several core techniques in J2EE
- Two APIs to process XML:
  - Document Object Model (DOM): a tree-oriented model
  - SAX (Simple API for XML): a stream-based event-driven processing model
- Java API for XML Binding (JAXB): mapping XML to and from Java classes

Web Services

- A web service is a software function that:
  - Has its interface be public
  - Can be called by other services or programs
- A business service is often designed and implemented by a web service
- Web services become a new method to develop softwares
- Service-Oriented Architecture (SOA): a software architecture that is based on web services

Transaction Support

- J2EE—and EJB in particular—provides substantial transaction support.
- The EJB container provides built-in support for managing transactions, and allows the developer to specify and modify transaction boundaries without changing code.
- Where more complex transaction control is required, the EJB can take over the transaction control from the container and perform fine-grained or highly customized transaction handling.

Security

- J2EE provides strong built-in security mechanisms
- Authorization in J2EE is based on roles of users of applications

JNDI

- Java Naming and Directory Interface
- Utilized by J2EE applications to locate resources and objects in portable fashion
  - Applications use symbolic names to find object references to resources via JNDI
  - The symbolic names and object references have to be configured by system administrator when the application is deployed.
- JNDI provides methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes.

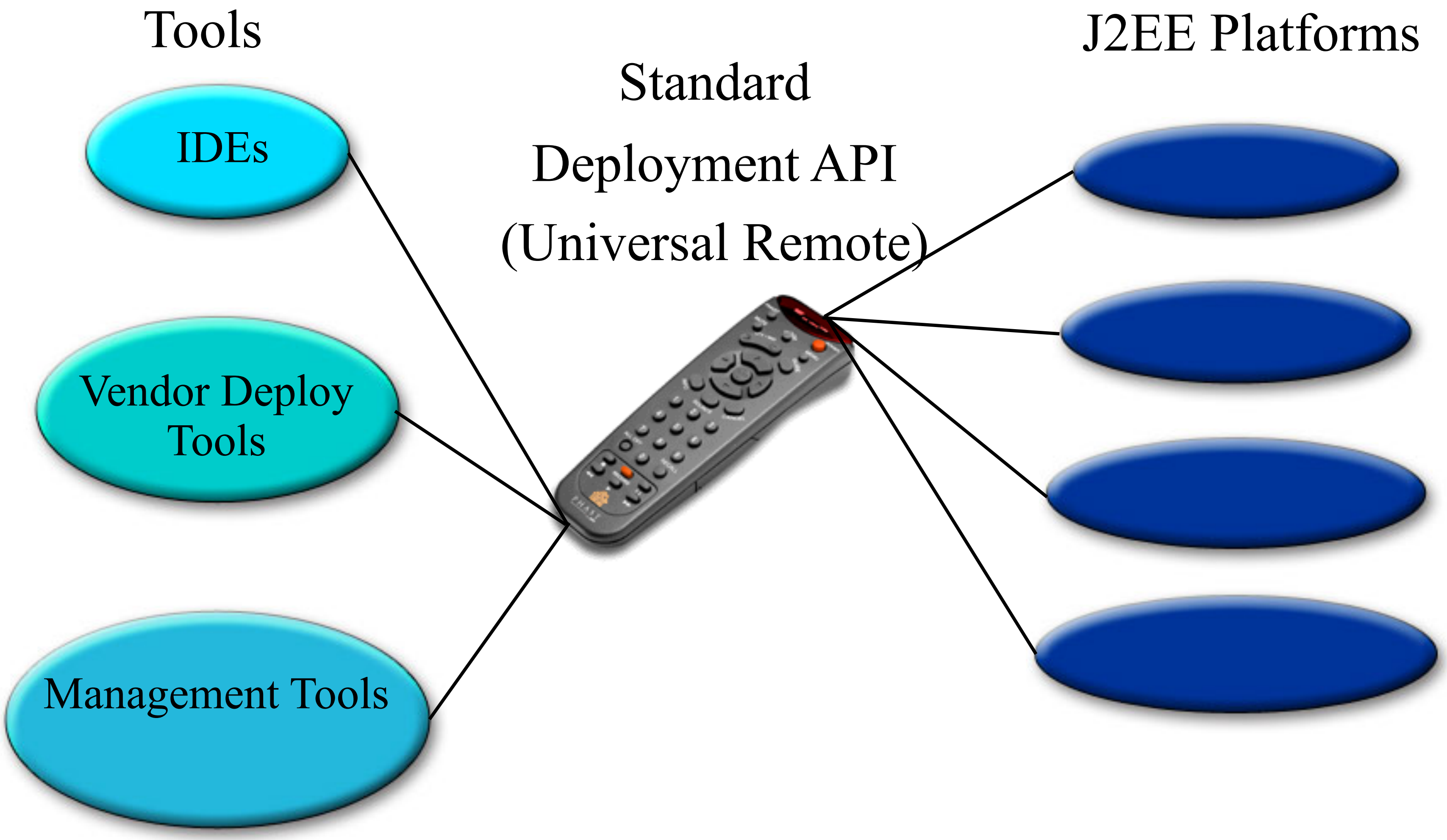
JDBC

- Provides standard Java programming API to relational database
  - Uses SQL
- Vendors provide JDBC compliant driver which can be invoked via standard Java programming API

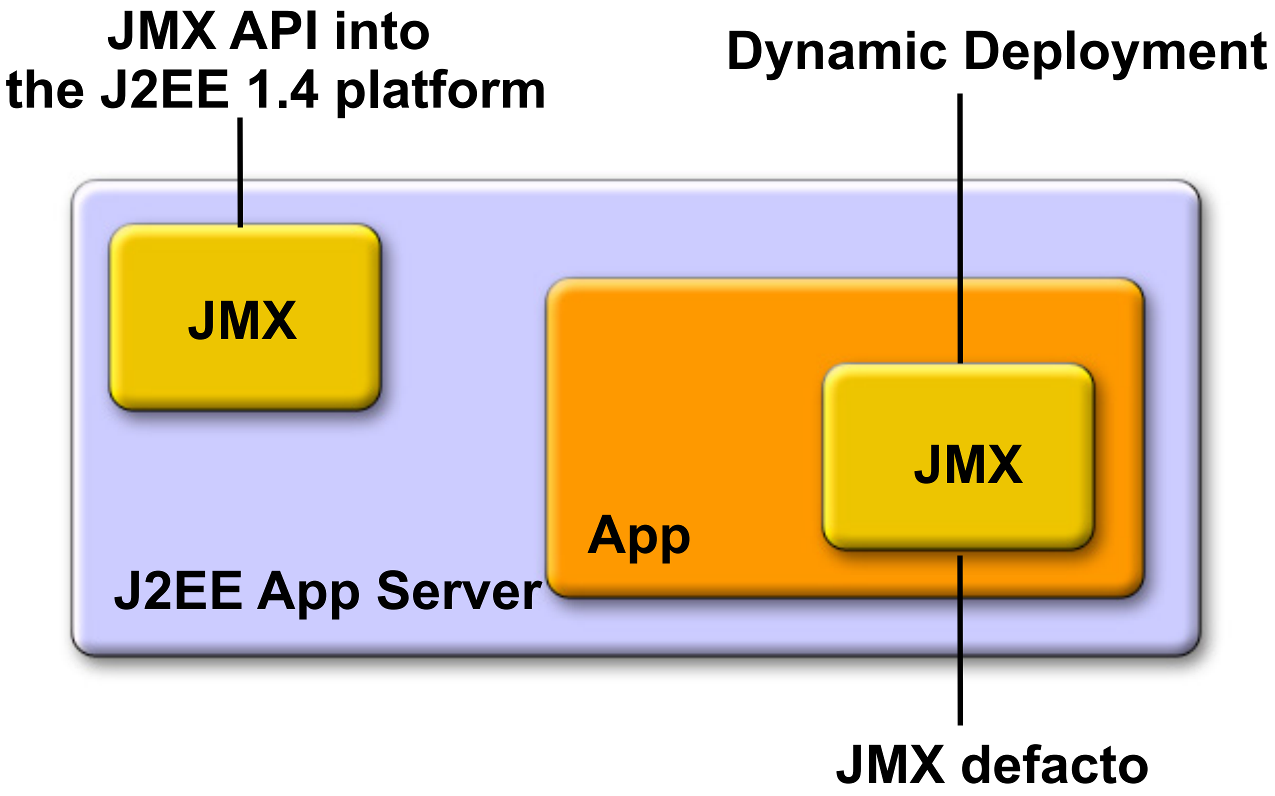
J2EE Management (JSR-77)

- Management applications should be able to discover and interpret the managed data of any J2EE platform
- Single management platform can manage multiple J2EE servers from different vendors
- Management protocol specifications ensure a uniform view by SNMP and WBEM management stations
- Leverages JMX

J2EE Deployment (JSR-88)



JMX

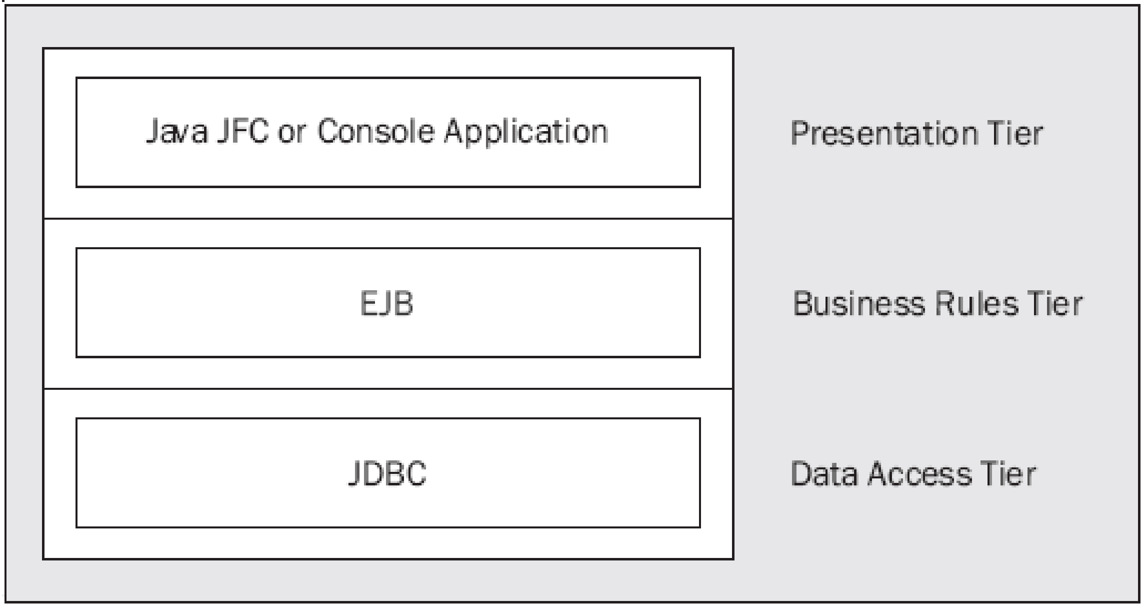


A single technology for the J2EE platform

Sample J2EE Architectures

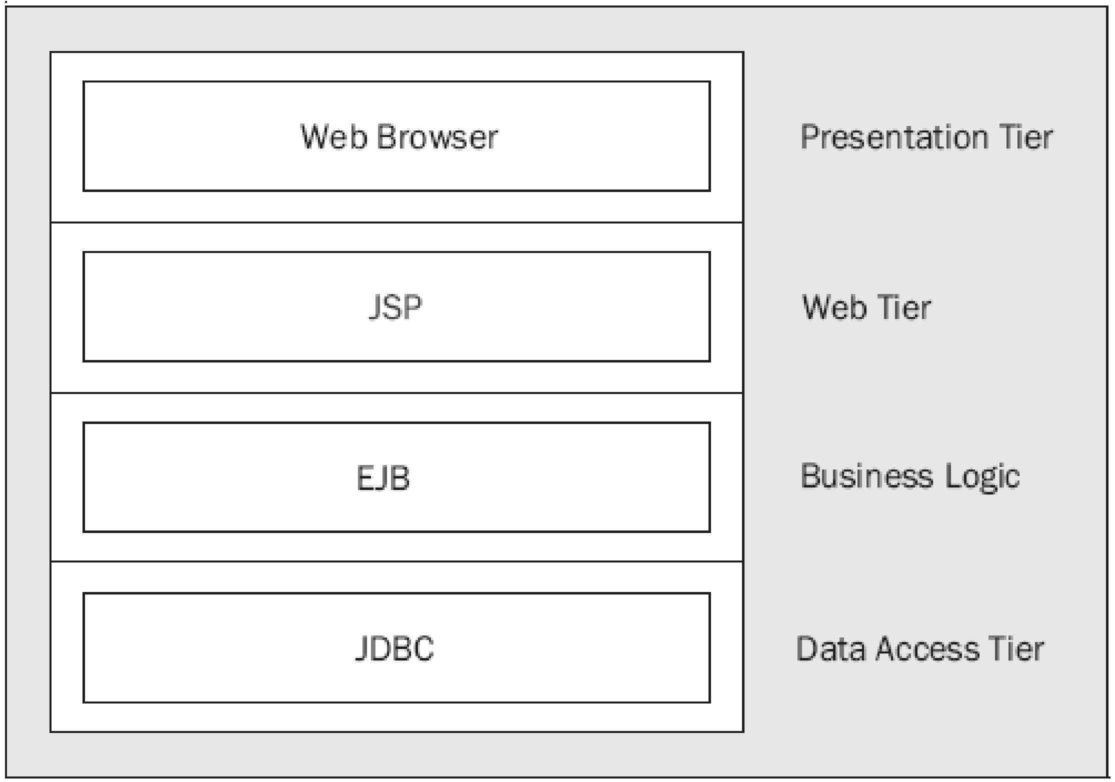
- Several different architectures for different types of applications, but some commons
- n-Tier Architecture is intended to solve:
  - High cost of maintenance when business rules change
  - Inconsistent business rule implementation between applications
  - Inability to share data or business rules between applications
  - Inability to provide web-based front ends to line-of-business applications
  - Poor performance and inability to scale applications to meet increased user load
  - Inadequate or inconsistent security across applications

Application Client with EJB



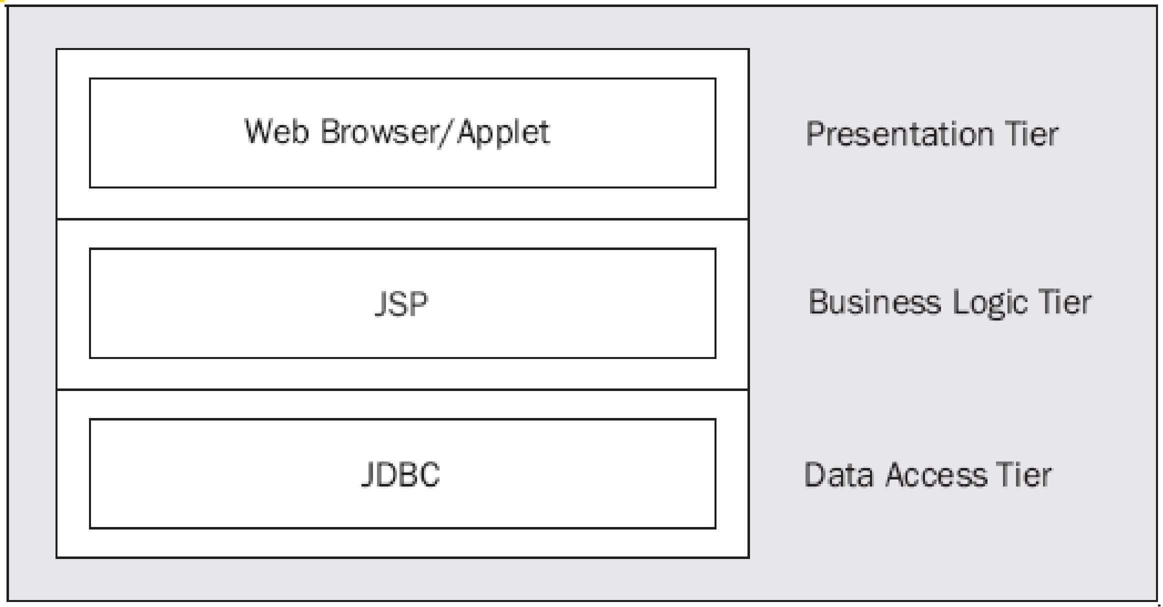
- The client application is built as a stand-alone (JFC/Swing or console) application.
- The application relies on business rules implemented as EJBs running on a separate machine.

JSP Client with EJB



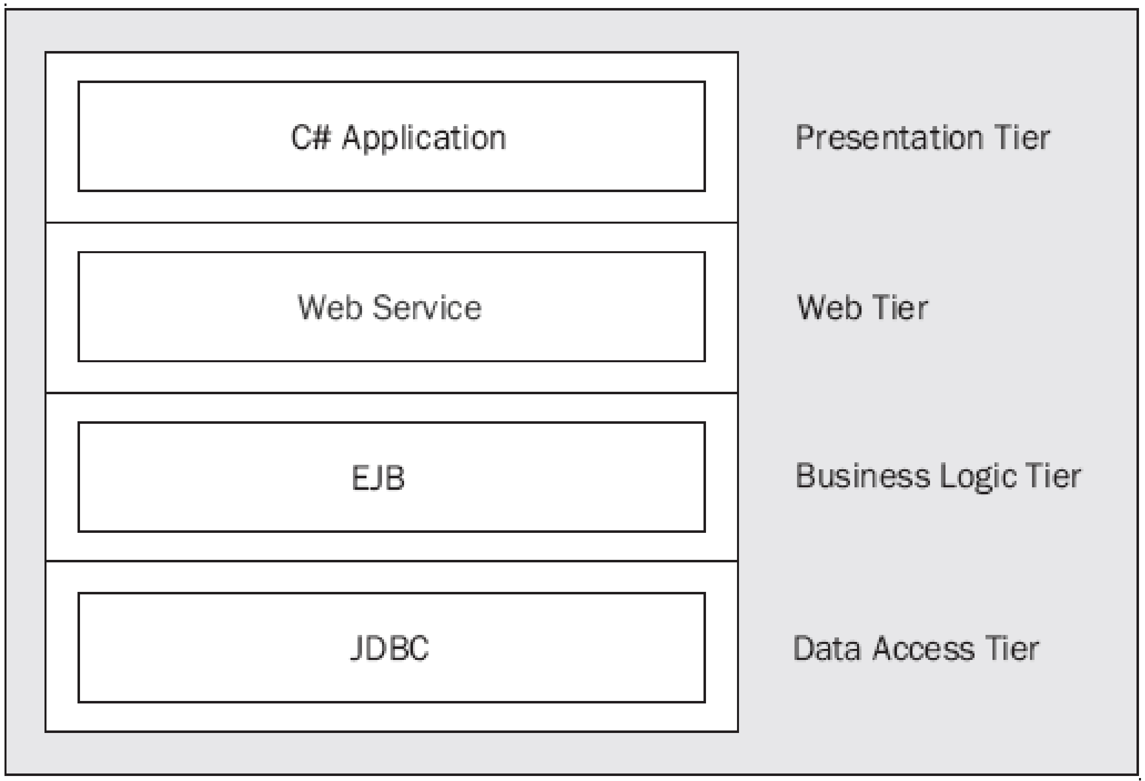
- The client in this architecture is a web browser.
- JavaServer Pages access business rules and generate content for the browser.

Applet Client with JSP and Database



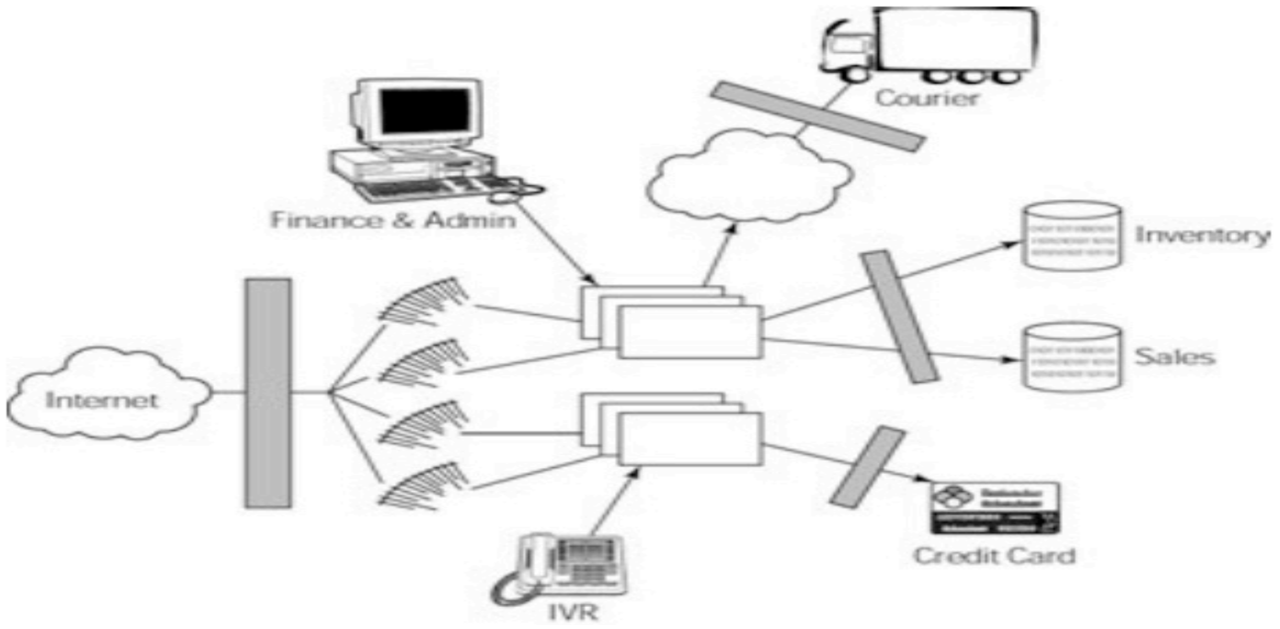
- The client application is a web browser, but in this case a Java applet is used within a web page to provide a more interactive, dynamic user interface for the user. That applet accesses additional content from JSPs.
- Data is accessed from the JSP via the JDBC API.

Using Web Services for Application Integration



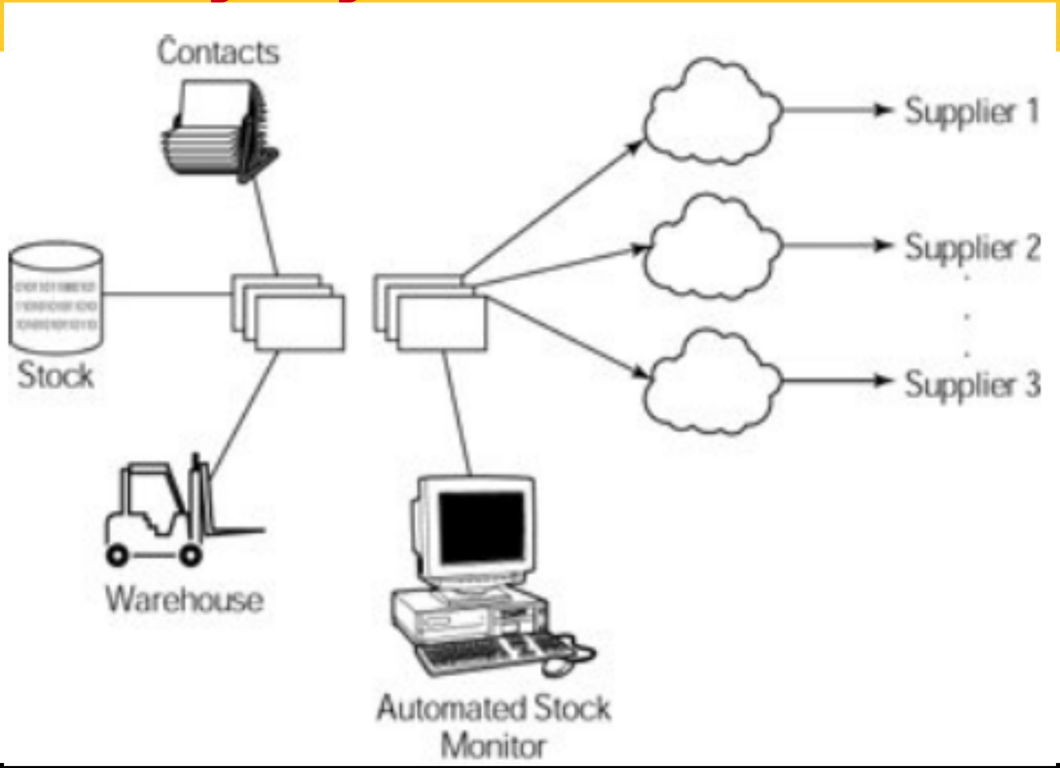
- A client application implemented in C# accesses data from a web service implemented in Java.

Sample J2EE Application 1 - B2C E−commerce Web site



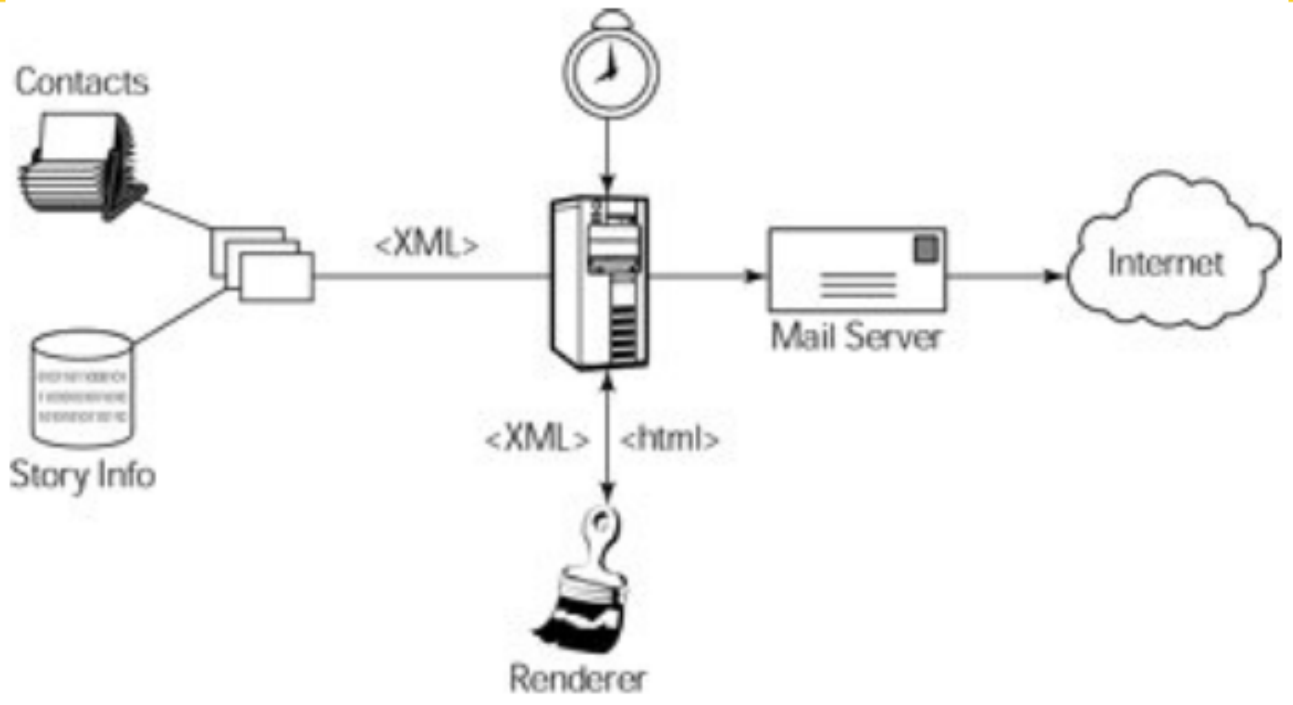
Web−Page Generation	Middleware	Database Server	Operating System
ASP	COM/DCOM	Usually SQLServer, but could be Oracle/Informix/Sybase	Microsoft
Servlet/JSP	EJB or CORBA	Oracle/Informix/Sybase	UNIX/Microsoft
CGI	CORBA	Oracle/Informix/Sybase	UNIX

Sample J2EE Application 2 - An Inventory System in B2B E–commerce



API	Use
EJB	Abstraction of business logic.
XML	Exchange of parts information and orders.
JNDI	Customer and supplier directory handling.

Sample J2EE Application 3 - Monthly electronic newsletter



API	Use
JavaMail	Interface to e−mail system.
XML	Stores formatted message information.
JDBC	Extracts address information directly from the database.

Homework

- ❖ Research online about J2EE
- ❖ Explore J2EE Architecture
- ❖ Application of J2EE in Web-App