CS-25 ADVANCE JAVA PROGRAMMING (J2EE)

HANDOUT MATERIAL

BCA 5TH SEM.

2018-19

Prepared By: Dr. Sharon V. Mohtra
Assistant Professor,
Dept. of Computer Applications,
Christ College,
Rajkot

J2EE Platform Introduction & Technologies

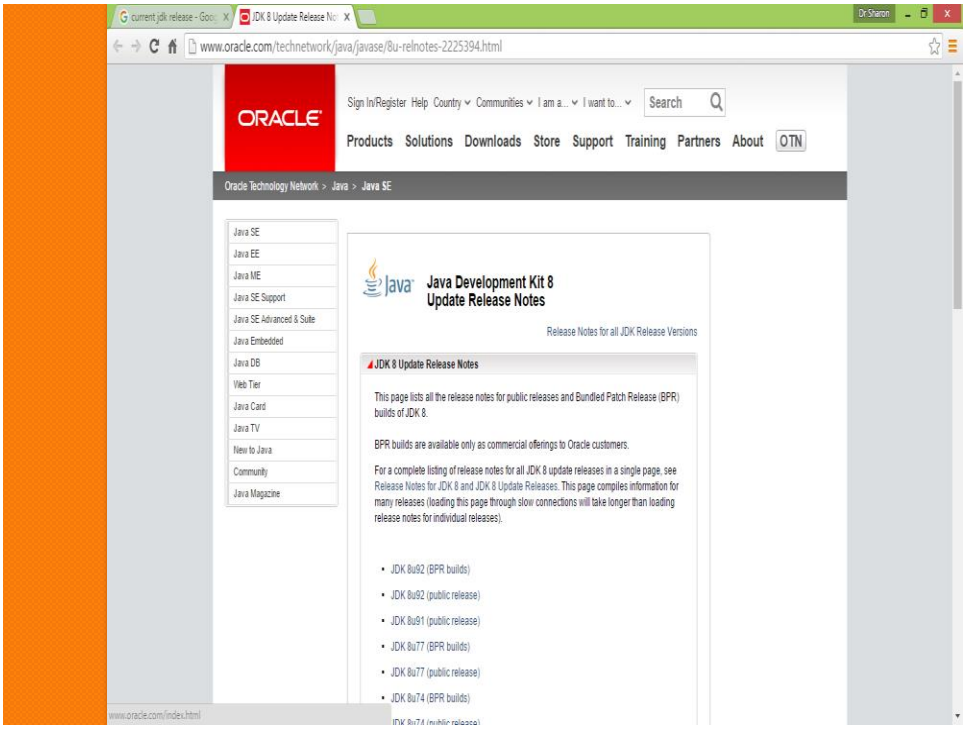
CS - 25 Advance Java Programming
Unit 1
Marks : 14

What is a language?

- J2EE ====technology, standards

- Java =====language

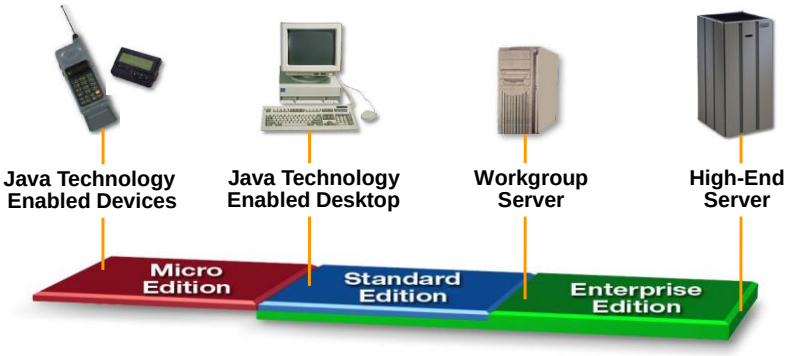
Variables,
Methods,
Classes,
Objects



Java

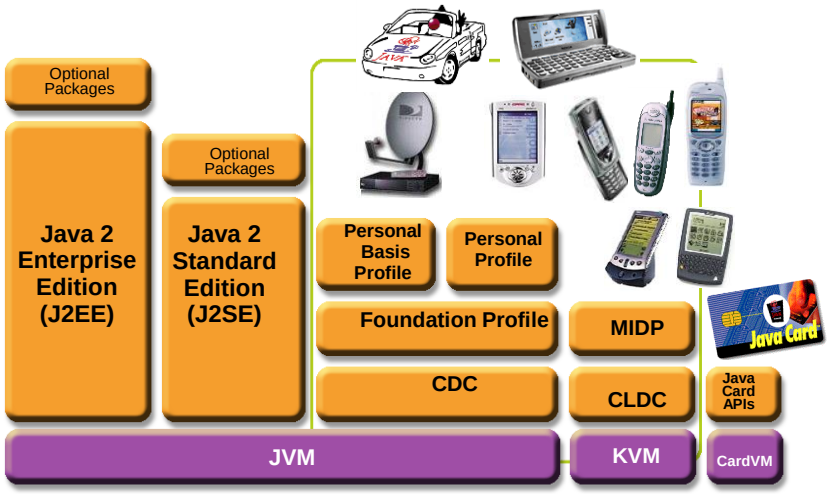
- First implementation released in 1995 (1.1)
- Java compiled to bytecode -> JVM
- Basic Philosophy
  - Object-oriented
  - Platform independent
  - Automatic memory management
- Current version 8+ (JDK)

The Java™ Platform



The Java™ Platform

Java 2 Platform Micro Edition (J2ME™)



Java EE → The next destination…

At a Glance

- Java Platform, Enterprise Edition (Java EE) is the **standard in community-driven enterprise software**. Java EE is developed using the **Java Community Process,** with contributions from industry experts, commercial and open source organizations, Java User Groups, and countless individuals.
- Each release integrates new features that align with industry needs, improves application portability, and increases developer productivity.
- Today, Java EE offers a rich enterprise software platform, and with **over 20 compliant Java EE 6 implementations** to choose from, low risk and plenty of options.

Overview of Java ™ Enterprise Edition

- Java EE platform is designed to help developers
  - **To create large-scale, multi-tiered, scalable, reliable, and secure network applications.**
- A shorthand name for such applications is "enterprise applications," That's why **Enterprise Edition.**
- This applications are not only useful for large corporations, agencies, and governments, but also essential for individual developers and small organizations in an increasingly networked world.
- The Java EE platform is designed to reduce the complexity like security and reliability of enterprise application development by providing **a development model, API, and runtime environment.**

What Is the JavaEE?

- Open and standard based platform for
- developing, deploying and managing
- n-tier, Web-enabled, server-centric, and component-based enterprise applications
- Versions:
  - J2EE 1.2 (December 12, 1999)
  - J2EE 1.3 (September 24, 2001)
  - J2EE 1.4 (November 11, 2003)
  - Java EE 5 (May 11, 2006)
  - Java EE 6 (December 10, 2009)
  - Java EE 7 (April 5, 2013)
  - Java EE 8 (August 31, 2017)

What J2EE Means...

- It is an architecture specification to provide/support distributed, multi-tiered, component based platform/infrastructure to enable developers to create (develop) and deploy enterprise applications that are secure, distributed, interoperable and scalable.
- It simplifies enterprise application development and maintenance by basing them on standardized, modular components, by providing a complete set of services to those components, and by handling details of application behavior automatically.

What J2EE Means...

- It's a separate edition of Java optimized to meet the usage and performance requirements of enterprise solutions.
- J2EE is *J2SE + additional API's* (that provide enterprise computing capabilities) and an execution environment for them.
- It is designed for construction of distributed apps and is based on MVC (Model-View-Controller) paradigm, a proven approach for designing multi-tier, enterprise apps that are scalable and maintainable.

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

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 to 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

Terms

1 web server: Web Server is software which serves http requests.

2 application server: Application server is a software which creates java request & response object on the base of http request & calls corresponding servlets or JSP.

3 deployment descriptor: set of xml files, mainly web.xml.

4 POJO: Plain Old Java Object. (stored & executed at same machine – client or server)

5 Applet: Applet is an application which can travel through network.(Stored at server, comes to client machine with html file & executed in client machine in browser)

6 Servlet: Servlet is an application is stored at server and executed by application server on http request & generates output as http response.(html viewed in browser on client machine.) (Servlet may forward, redirect request to other servlets or jsp. Servlet may include output of other servlet or jsp into its own output.)

7 Jsp: JSP stands for Java Server Page. JSP page has extension .jsp which may contains java code(jsp tags / java objects) inside html.

8 Client: Software which makes request to server is called client-known as browser.

9 Request: Data sent from client to server is called request, which describes requirement of client to server.

10 Response: Data sent from server to client as reply to request is called response.

11 query string: String format of request is called query string.

12 url: Universal resource locator.

13 Parameter: Variable/field passed from client to server, Which can not be modified by server.

14 Attribute: Variable/field setted by server-component and later accessed by server-component, is called attribute which can be modified/removed by server.

15 Cookie: Small piece of text information stored in client machine by server to track client.

16 Session: Sequence of transactions stored on server to track client.

17 Apache-jakarta-tomcat Apache is group. Jakarta is one of the projects of apache group. Tomcat is one of the freeware products developed by apache group under Jakarta project.

18 IDE IDE stands 4 Integrated Development Environment. There are so many IDE 4 Java developed by third party like edit-plus, NetBeans, Eclipse, Oracle JDeveloper, Borland Jbuilde, etc…

Enterprise Computing

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

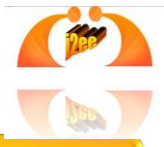
Key Technologies
J2SE™
J2EE™
- JMS
- Servlet
- JSP
- Connector

XML
- Data Binding
- XSLT (Extensible Stylesheet Language Transformation)

Products
App Servers
Web Servers
Components
Databases
Object to DB tools

Legacy Systems
Databases
TP Monitors
EIS Systems

Java EE is made up of…..



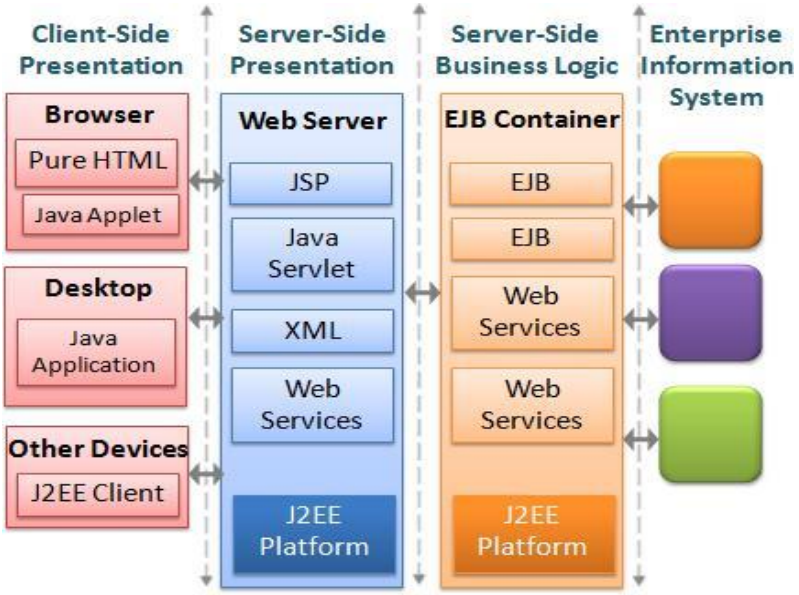
- API and Technology specifications
- Deployment and Development platform
- Standard and Production Quality implementation
- Compatibility Test Suites (CTs)
- J2EE Brand ™
- J2EE Blueprint
- Sample codes

Evolution of Enterprise Application Development

About Enterprise Application

- Things that make up an enterprise application
  - - Presentation logic
  - - Business logic
  - - Data access logic (and data model)
  - - System services
- The evolution of enterprise application framework reflects
  - - How flexibly you want to make changes
  - - Where the system services are coming from

Enterprise Architecture styles



Enterprise Application Architecture Models

- Single tier
- Two tier
- Three tier
- N- tier

The Tier Architecture

- Two – Tier Architecture
- Three Tier Architecture
- N – Tier Architecture
- Enterprise Architecture

Basic Terminology of EAS

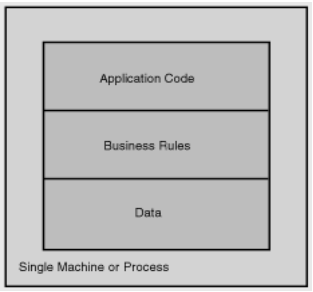
- **Layers**:-Logical sepration of an application
- There are basic 6 layers

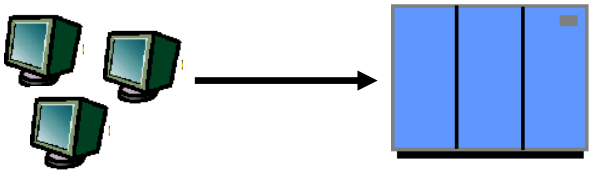
Presentation Manager	**Presentation Logic**
Application Logic	**Business Logic**
Database Logic	**Database Manager**

- **Tiers:-** Physical sepration of an application
- There are basic 3 tiers
  - Client Tier/ Presentation tier
  - Middle Tier/ Server
  - Database Tier

Single Tier Architecture

- Dumb terminals are directly connected to mainframe
- Centralized model (as opposed distributed model)
- Presentation, business logic, and data access are intertwined in one monolithic mainframe application
- Eg. FOXPRO

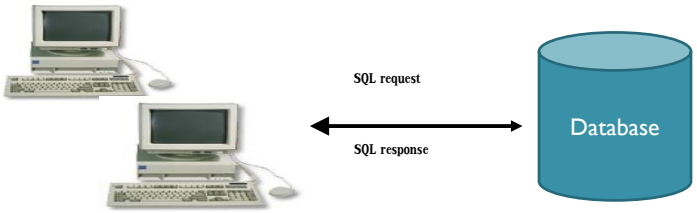




Single Tier Architecture (Pros. & Cons.)

- Pros:
  - Easy to manage
  - No client side management is required
  - Data consistency is easy to achieve
- Cons:
  - Can't share data in large amount
  - Functionality (presentation, data model, business logic) intertwined, difficult for updates and maintenance and code reuse

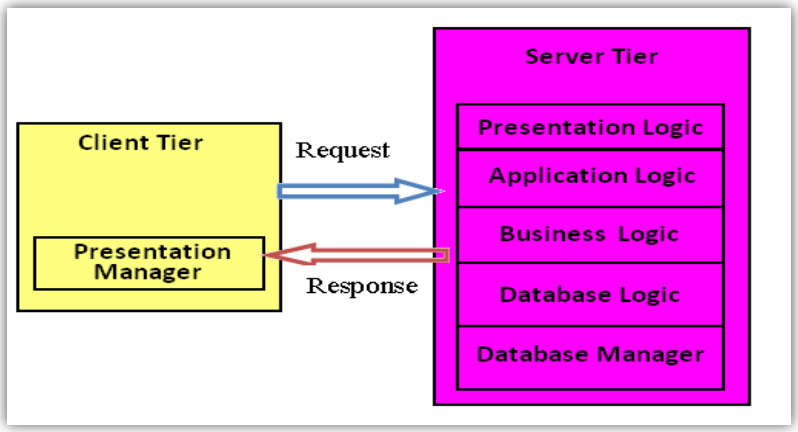
Two Tier Architecture



- Clients talking to back end database
  - SQL queries sent, raw data returned
- Arrangement of layers is according to type of Client (Thin client, Thick client & Normal client)
- Presentation, Business logic and Data Model processing logic in client application
  - Client – Server systems
  - Ex:- Unix, mainframes

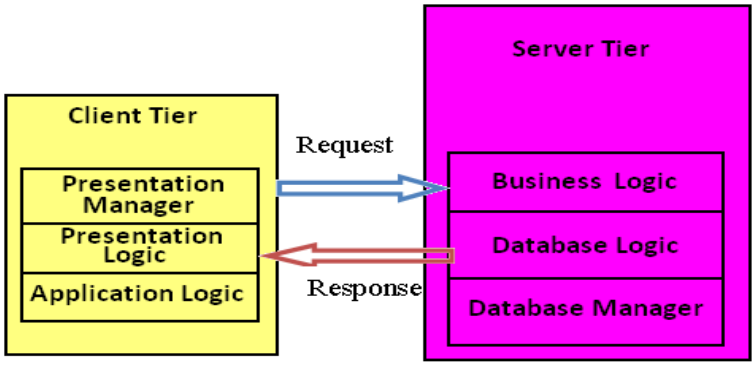
Two Tier: Thin Client

- Presentation Manager resides only with the client is called Thin client.
- Other layers reside with the Server side/ Database tier



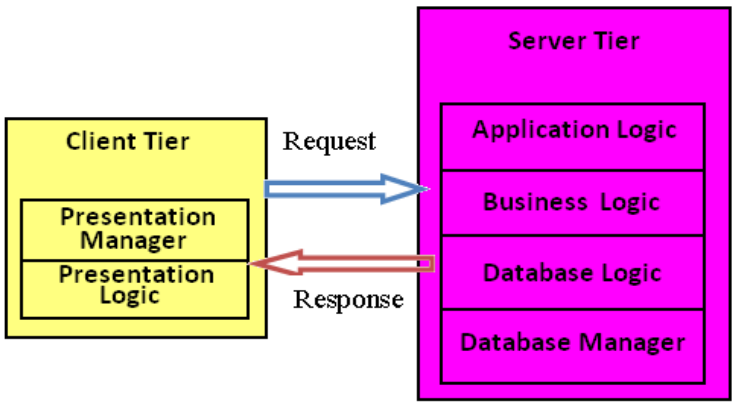
Two Tier: Thick Client

- Presentation Manager, Presentation logic and Application logic reside with the client tier is called Thick client.
- Other layers reside with the Server side/ Database tier



Two Tier: Normal Client

- Presentation Manager and Presentation logic reside with the client tier is called Thick client.
- Other layers reside with the Server side/ Database tier



Two Tier (Pros. & Cons.)

- Pros:
  - DB product independence (compared to single-tier model)
  - Any changes made in data access logic will not affect presentation and business logic.
  - With the two tier architecture it is easy to develop an application.
- Cons:
  - An application is expected to support a limited number of users. The reason is that each client requires its own connection and each connection requires CPU and memory. So, as the number of connections increases, the database performance degrades.
  - Presentation, data model, business logic are intertwined (at client side), difficult for updates and maintenance
  - Data Model is “tightly coupled” to every client: If DB Schema changes, all clients break
  - Updates have to be deployed to all clients making System maintenance nightmare
  - DB connection for every client, thus difficult to scale
  - Raw data transferred to client for processing causes high network traffic

Three Tier Architecture

- Thinner client: business & data model separated from presentation
  - - Business logic and data access logic reside in middle tier server while client handles presentation
- Middle tier server is now required to handle system services
  - - Concurrency control, threading, transaction, security, persistence, multiplexing, performance, etc.

Three Tier (RPC based): (Pros. & Cons.)

- Pros:
  - Business logic can change more flexibly than 2-tier model Most business logic reside in the middle-tier server
- Cons:
  - Complexity is introduced in the middle-tier server
  - Client and middle-tier server is more tightly coupled (than the three-tier object based model)
  - Code is not really reusable (compared to object model based)

Three-tier (Web Server based): Pros & Cons

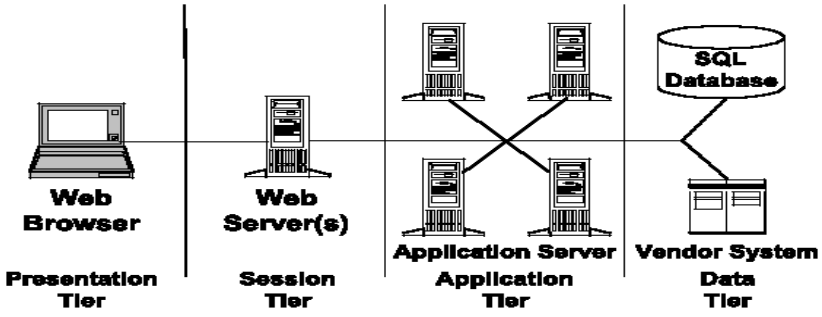
- Pro:
  - Ubiquitous client types
  - Zero client management
  - Support various client devices
    - J2ME-enabled cell-phones
- Cons:
  - Complexity in the middle-tier still need to be addressed

Trends

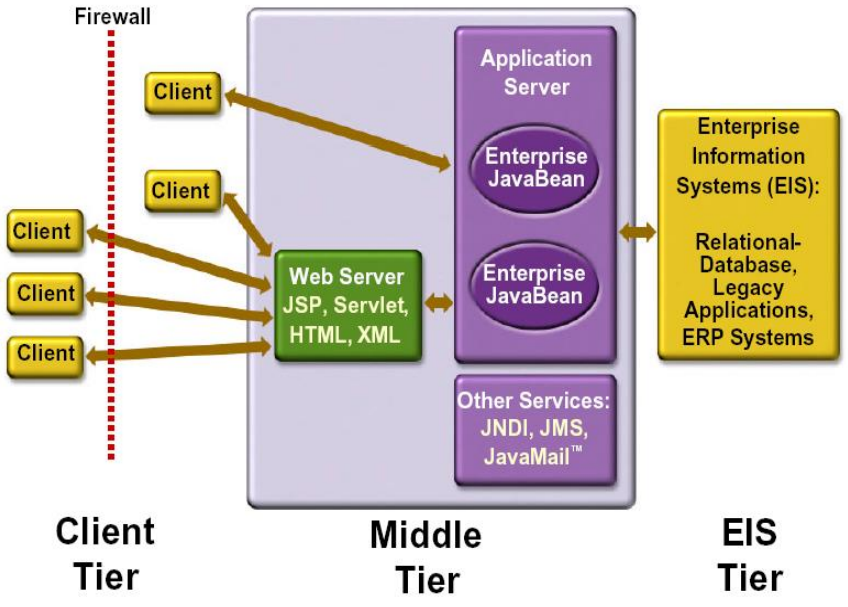
- Moving from single-tier or two-tier to multi-tier architecture
- Moving from monolithic model to object-based application model
- Moving from application-based client to HTML-based client

N Tier

- N-Tiered architecture: Various components that make up the application are logically separated or distributed across network.
  - Client ←→ Server ←→ Server ←→ Database
  - Eg. ATM Application



J2EE is End-to-End Solution



Tier Architecture in J2EE

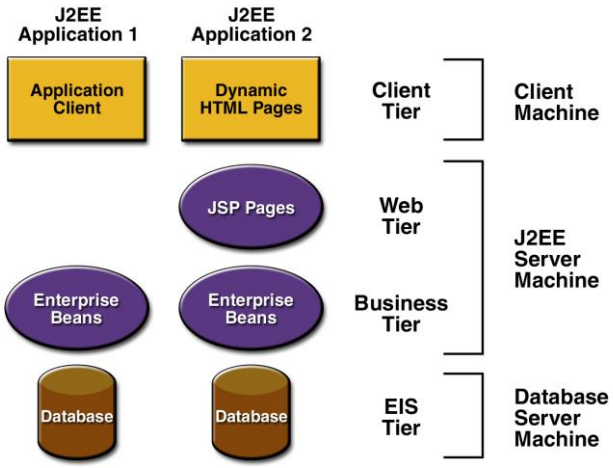
- Front-end (Client):
  - Viewed and manipulated by the users.
  - It can live in a Web browser or a standalone application.
  - Presents customized information to clients requirements.
  - Servlets and JSP is used as Front end development.

Tier Architecture in J2EE

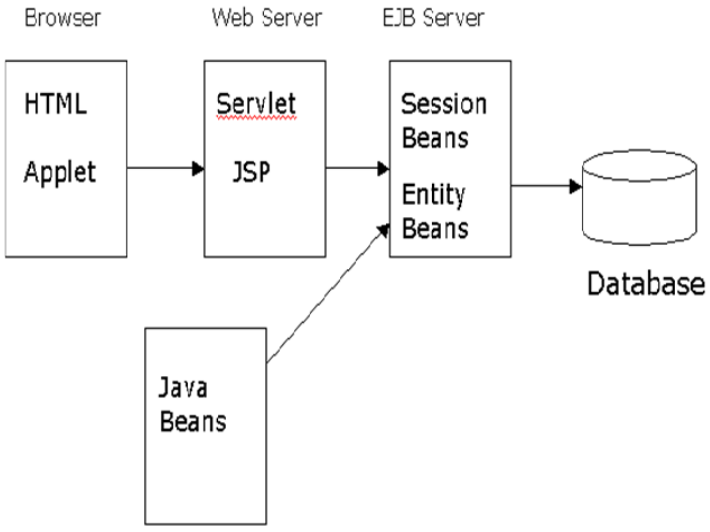
- Middle:
  - Contains business logic Ex: Discounts.
  - It may contain two sub-tiers:
    - Web Tier – It handles communication to client.
    - EJB Tier – It manages business logic and access to corporate data.

- Backend (EIS):
  - Provides access to various corporate data stores (Databases, E-Mail system, Legacy systems…)

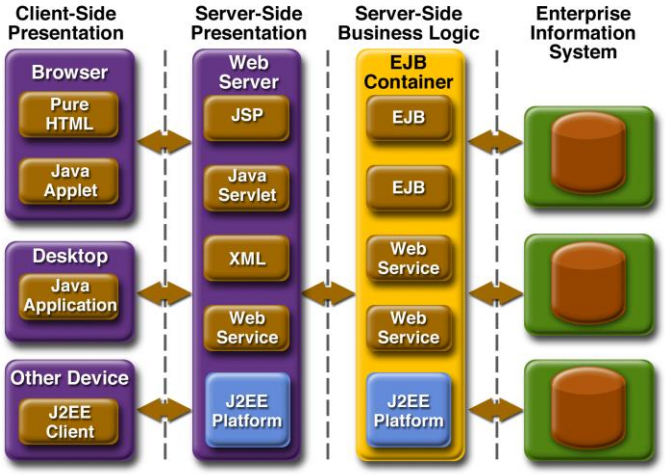
Tier Architecture in J2EE



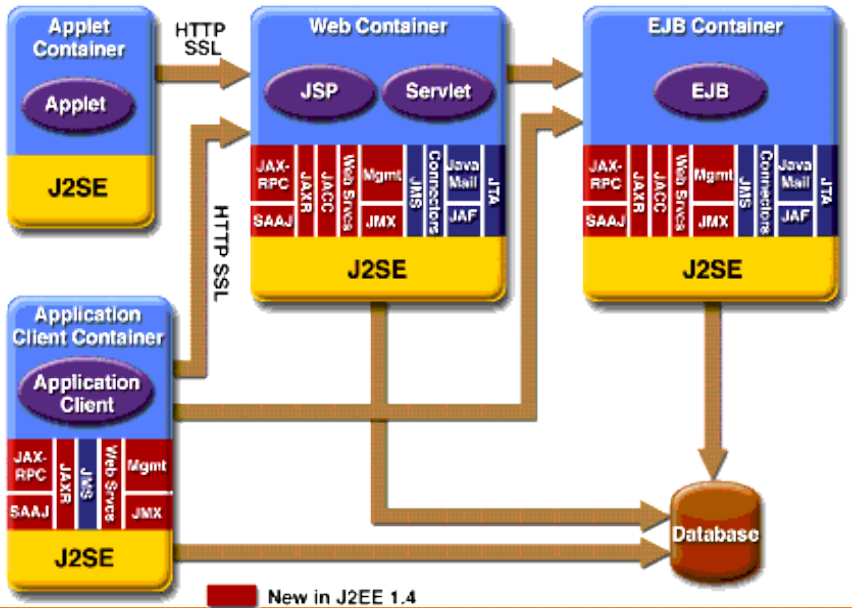
J2EE Application Model



The J2EE Platform (EAS Model)



The J2EE Platform (APIs)



J2EE 1.4 APIs and Technologies

- J2SE 1.4 (improved)
- JAX-RPC (Java API for XML-based **RPC)** (new)
- Web Service for J2EE
- J2EE Management
- J2EE Deployment
- JMX (Java Management Extensions)1.1
- JMS (Java Message Service)1.1
- JTA (Java Transaction API)1.0
- Servlet 2.4
- JSP 2.0
- EJB 2.1
- JAXR (Java API for XML Registries)
- Connector 1.5
- JACC (Java Authorization Contract for Containers)
- JAXP (Java API for XML Processing) 1.2
- JavaMail 1.3
- JAF (Java beans Activation Framework)1.0

Need for J2EE

- Integration with legacy EIS.
- Reliability and confidence.
- Complexity (mission-critical) in building and maintaining.
  - Modules exist on heterogeneous environment.
  - Different modules written in different languages.
  - Different versions of the same software may exist.

J2EE Goals

- To provide an architecture to:
  - Reduce server down-time.
  - Increase application scalability (raise to demands).
  - Application stability (should execute as expected without crashing without exhibiting buggy nature or incompleteness).
  - Secure (be tolerant to unauthorized access of vital data).
  - Simplicity (to enable faster development and maintenance).

Advantage of J2EE

- Being developed and enhanced under the ordinance of JCP, meets the real-time requirements of enterprise application developers.
- Portable deployment – “develop once, deploy anywhere” mantra.
- Forces to abide by three-tiered architecture and supports n-tier.

Advantages of J2EE

- Provides infrastructure/design to enable developers to create, distributed and interoperable enterprise apps.
- It is a distributed, multi-tiered and component based architecture that facilitates scalable applications.
- Scalability of not only applications performance but also of application development process.

Reason for using J2EE ?

- To provide fast, reliable access to corporate databases from the Web.
- To build dynamic, data-driven web applications for large user populations that expect 24*7 availability.
- To automate E-Mail or wireless communications with partners, vendors, employees or customers.
- To implement complex business logic.

Reason for using J2EE ?

- To provide robust user authentication/authorization for web resources and other services.
- To write applications that seamlessly integrate data from disparate sources on multiple platforms.
- To execute distributed transactions across multiple data stores.

Intro. To J2EE APIs

- **Servlet**
- Java Servlets are the Java equivalent of CGI scripts that can be used to perform processing and the servicing of client requests on a web server.
- Servlets are simply Java classes that implement a predefined interface.
- They can be used to dynamically generate content for presentation to the user, and this is achieved by embedding markup language (e.g. HTML) inside the Java code.
- They have access to the rich library of features provided by Java, including access to databases and other enterprise resources such as EJB.

- **JSP**
- JSP is another technology for presenting information to the user over the web and uses a paradigm where Java code is embedded into the HTML - the opposite of servlets, and much like Microsoft ASP. Pages are written as HTML files with embedded Java source code known as scriptlets.
- One of the pitfalls in using JSP is that it is very easy to build large pages containing lots of embedded Java code and business logic. For this reason, JSPs provide easy integration with JavaBeans and another feature called JSP tag extensions. These custom tags (also known as custom actions) allow re-usable functionality to be encapsulated into XML-like tags that can be easily used on the pages by both page developers and designers.

- **EJB**
- EJB is a major part of the J2EE specification and defines a model for building server-side, reusable components. There are three types of enterprise beans currently supported by J2EE - session beans, entity beans and message-driven beans.
- Session beans can be seen as extensions to the client application and are typically used to model business processes. There are two types of session bean - stateful and stateless. Stateful session beans are typically used to record conversational state for a single client between requests, whereas stateless session beans are shared between any number of clients at any one time.

- Entity beans are typically used to model persistent business entities and, in particular, data in a database. A common mapping is to model an entity bean on a table, there being one instance of that bean for every row in the table. There are two ways that persistence can be achieved - container managed and bean managed persistence.
- In container managed persistence, a mapping is defined at deployment time between the persistent properties in the bean and the columns in the table. With bean managed persistence, developers write the JDBC code that performs the create, read, update and delete operations.
- Finally, message-driven beans allow functionality to be executed on an asynchronous basis, typically triggered by JMS messages from message-oriented middleware.

- **JMS**
- JMS is Java API that presents an interface into message-oriented middleware such as IBM MQSeries, SonicMQ and so on.
- Like JDBC, JMS provides Java applications a mechanism to integrate with such systems by presenting a common programming interface irrespective of the underlying messaging system.
- Functionally, JMS allows messages to be sent and received using a point-to-point or publish/subscribe paradigm.

- **JavaMail**
- J2EE applications use the JavaMail API to send email notifications. The JavaMail API has two parts: an application-level interface used by the application components to send mail, and a service provider interface. The J2EE platform includes JavaMail with a service provider that allows application components to send Internet mail.

- **JSF**
- JavaServer Faces (*JSF*) is a Java specification for building component-based user interfaces for web applications.

- **JNDI** (Java Naming and Directory Interface)
- The Java Naming and Directory Interface (JNDI) provides naming and directory functionality. It provides applications with methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes. Using JNDI, a J2EE application can store and retrieve any type of named Java object.

Apache Tomcat as a Web Container

- Apache Tomcat (or simply Tomcat, formerly also *Jakarta Tomcat*) is an open source web server and servlet container developed by the Apache Software Foundation (ASF).
- Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run in.
- Apache Tomcat includes tools for configuration and management, but can also be configured by editing XML configuration files.

J2EE 1.4 as an Application Server

- The platform incorporates a design based largely on modular components running on an application server.
- Software for Java EE is primarily developed in the Java programming language.

Small steps lead to a big leap…

….