**CH-1 The J2EE Platform**

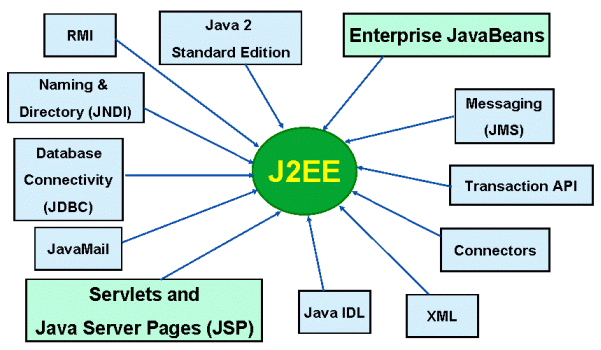
* **Introduction to J2EE**
* J2EE Stands For Java 2 Platform Enterprise Edition.
* J2EE is a Plate Form independent.
* J2EE is a specification from Sun Microsystems for standardizing the implementation

of distributed, component based, java based enterprise applicationsJ2EE is a suite

of specifications each of which addresses a specific technology targeted towards

solving a specific problemJ2EE provides standards **for designing, developing and deploying enterprise applications**.

* More Then One Way to do Things.
* You Need Prior Knowledge of Java.
* You Need To know Little HTML.
* You Need To have idea about XML.
* You Need To have Idea about SQL.
* J2EE technology provides a component-based approach to the design,development,assembly,and deployment of enterprise applications.
* The J2EE platform provides the ability to reuse components in different applications.
* J2EE is made up of 13 different technologies including JavaServer Pages,Servlets,Enterprise JavaBeans,JavaMail,XML,JavaMail and many more .



* **Enterprise Architecture types / Styles**

**Tiers:**

A tier is an abstract concept that defines a group of technologies that provides one or more services to its client.

Enterprise architecture is divided in four type such as,

1)Single Tier Architecture

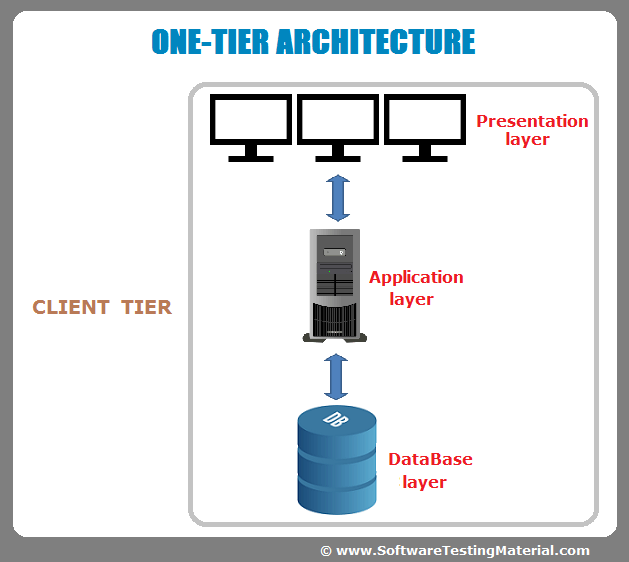
2)Two Tier Architecture

3)Three Tier Architecture

4)Multi Tier (n – tier)

1. **Single Tier Architecture**

* Single tier architecture is an architecture which the entire application rsides on the user’s machine.
* A Tier is Also referred to as “**Layer**”.
* All processing in a Single Computer.
* All Resource To the Attached to the same computer.
* A single-tier is not a distributed application. For example, Microsoft Word, Open Office Writer.



**Advantages :**

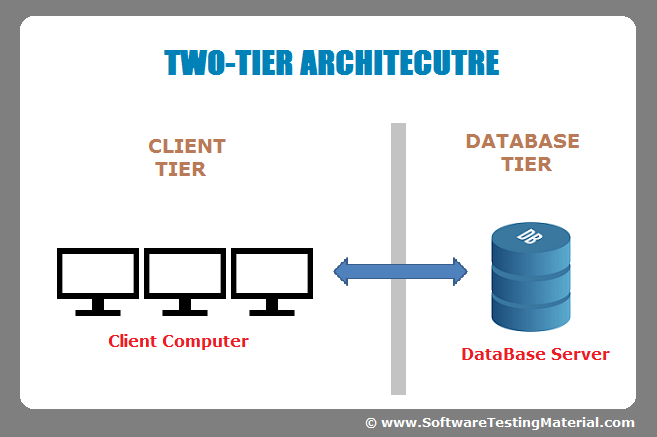
* Simple
* Efficient
* Uncompleted

**Disadvantages :**

* terribly expensive to run

**2)Two Tier Architecture**

* + Two Tier Architecture AKA Client server application.
  + Two Tier Architecture is divided into two parts.
    - Client Application(Client Tier)
    - Database(Data Tier)
  + Client System Handles Both Presentation And Application layers and server system handles Database layer.
  + It is also known as a client server Application.



**Advantages :**

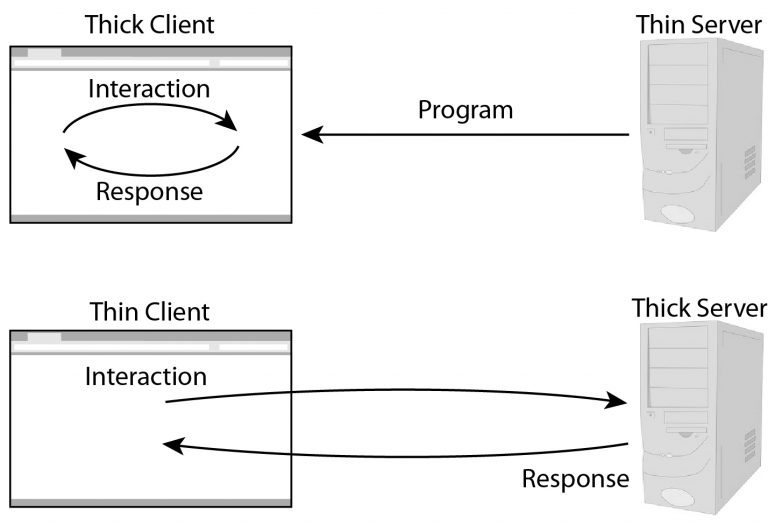
* Easy and Fast to setup
* Easy to administrate

**Disadvantages :**

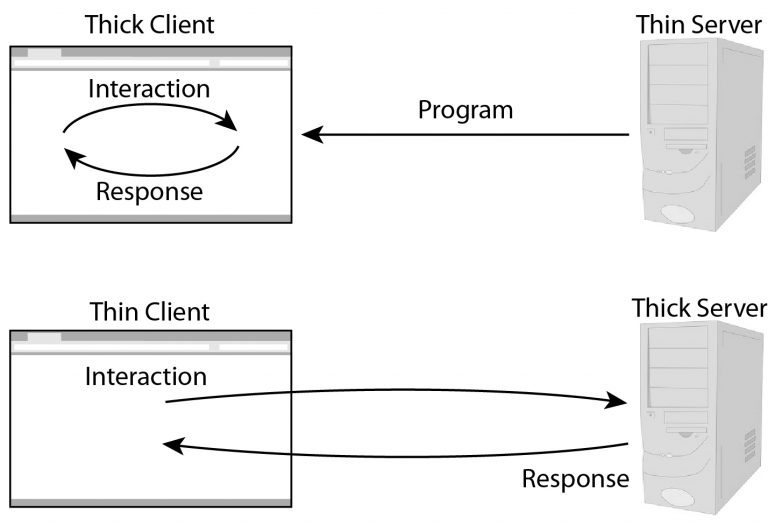
* Not Fail Safe(single point of failure)
* Scales badly on high loads.

|  |  |  |
| --- | --- | --- |
| Sr.No Factors | Thin Client | Thick Client |
| 1 Installation | Thin Clients Have Browser Based installation. | Thick Clients are installed locally. |
| 2 Type of Devices | Thin clients are used by handheld devices | Customization systems use thick clients |
| 3 Processing Type | In thin client there is complete processing on server side. | Thick clients make use of computer resources more than server. |
| 4 Deployability | Deployable as compared to thick clients. | Thick clients are more expensive to deploy. |
| 5 Data Validation | The verification is required from the server side. | The data verification is done by client side. |
| 6 Communication | In thin clients continuous communication is required from server side. | In thick clients communication is done at particular intervals with the server side. |

* **Thin Client :**
  + Thin Clients Have Browser Based installation.
  + Thin clients are used by handheld devices
  + **Benefits of Thin Clients.**
    - Less vulnerable
    - Longer lifecycles
    - Uses less power
    - Less expensive
    - More scalable



* **Thick Client:**
  + Thick client is also known as Fat Client.
  + Thick Clients are installed locally.
  + Customization systems use thick clients
  + **Benefits of Thick Clients**
    - Working offline
    - Server connection
    - Server capacity
    - More flexibility
    - Existing infrastructure
    - Storage



* **Normal Client :**
  + Normal client is just the intermediate from of the two extremes .Here in this form presentation manager and presentation logic resides at client tier
  + Others like application logic, business logic, database logic and database manager resides with server side.

**3)Three Tier Architecture**

* + Three Tier application AKA Web Based Application.
  + Typical examples of a thin client three – tier application are most web - based applications.
  + Typical examples of three tier applications are applications that use technologies as CORBA (*Common Object Request Broker Architecture)* and DCOM(Distributed Component Object Model).
  + The three tier architecture is divided into three parts:

1)Presentation Layer(Client Tier)

2)Application Layer(Business Tier)

3)Database Layer(Data Tier)



**Advantages :**

* Easy to optimization perform
* No context switching

**Disadvantages :**

* High Maintenance
* Not Scalable
* Hard to modify

1. **Multi Tier Architecture :** 
   * n tier architecture is also called **n – Tier.**
   * n tier architecture would involve dividing and application into three different tiers.
     + 1 )Logic tier
     + 2)Presentation tier
     + 3)Data tier
   * **Benefit of n tier Architecture**
     + Security
     + Easy to manage
     + Scalable
     + Flexible

* **Enterprise Architecture**
* The J2EE platform uses a multitiered distributed application model.
* 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 multitiered J2EE environment to which the application component belongs.
* Following figure shows two multitiered J2EE applications divided into the tiers described in the following list.

• Client-tier components run on the client machine.

• Web-tier components run on the J2EE server.

• Business-tier components run on the J2EE server.

• Enterprise information system (EIS)-tier software runs on the EIS server.

* J2EE multitiered applications are generally considered to be three-tiered applications because they are distributed over three different locations: client machines, the J2EE server machine, and the database or legacy machines at the back end.
* 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 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. The J2EE specification defines the following J2EE components:
* Application clients and applets are components that run on the client.
* Java Servlet and JavaServer Pagestm (JSPtm) technology components are Web

components that run on the server.

* Enterprise JavaBeanstm (EJBtm) components (enterprise beans) are business

components that run on the server.

* J2EE components are written in the Java programming language and are compiled in the same way as any program in the language.
* The difference between J2EE components and "standard" Java classes is that J2EE components are assembled into a J2EE application, verified to be well formed and in compliance with the J2EE specification, and deployed to production, where they are run and managed by the J2EE server.

### J2EE Clients

A J2EE client can be a Web client or an application client.

#### Web Clients

* A Web client consists of two parts: dynamic Web pages containing various types of markup language (HTML, XML, and so on), which are generated by Web components running in the Web tier, and a Web browser, which renders the pages received from the server.
* A Web client is sometimes called a *thin client*. Thin clients usually do not do things like query databases, execute complex business rules, or connect to legacy applications. When you use a thin client, heavyweight operations like these are off-loaded to enterprise beans executing on the J2EE server where they can leverage the security, speed, services, and reliability of J2EE server-side technologies.

#### Applets

* A Web page received from the Web tier can include an embedded applet. An applet is a small client application written in the Java programming language that executes in the Java virtual machine installed in the Web browser. However, client systems will likely need the Java Plug-in and possibly a security policy file in order for the applet to successfully execute in the Web browser.
* Web components are the preferred API for creating a Web client program because no plug-ins or security policy files are needed on the client systems. Also, Web components enable cleaner and more modular application design because they provide a way to separate applications programming from Web page design. Personnel involved in Web page design thus do not need to understand Java programming language syntax to do their jobs

**Application Clients**

* A J2EE application client runs on a client machine and provides a way for users to handle tasks that require a richer user interface than can be provided by a markup language.
* • It typically has a graphical user interface (GUI) created from Swing or Abstract Window Toolkit (AWT) APIs, but a command-line interface is certainly possible.Application clients directly access enterprise beans running in the business tier. However, if application requirements warrant it, a J2EE application client can open an HTTP connection to establish communication with a servlet running in the Web tier.

**j2ee Multitire architecture**

* The architecture of the J2EE platform uses a multi-tier distributed application model.The application logic according to the functions, is divided into different components and various application components make up J2EE application.This J2EE application is then installed on different machines depending on the tier of the multi-tier J2ee environment for the application component.
* While a J2EE application can consist of three or four tires, J2EE multi tier applications are generally considered to be 3 tier applications because the architecture defines client-tier, a middle tier and a backend tier. This means that the application logic is distributed over different locations-client machines, J2EE server machine and database or lagecy machine at the backend

The advantages of the multi-tier architecture are:

• Forced separation of user interface logic and business logic.

• Business logic sits on small number of centralized machines (may be just one).

• Easy to maintain, to manage, to scale, loosely coupled etc.

* Each tier is assigned a unique responsibility in a 3-tier system. Each tier is logically separated and loosely coupled from each other, and may be distributed as follow.



**Client tier** represents Web browser, a Java or other application, Applet, WAP phone etc. The client tier makes requests to the Web server who will be serving the request by either returning static content if it is present in the Web server or forwards the request to either Servlet or JSP in the application server for either static or dynamic content.

**Presentation tier** encapsulates the presentation logic required to serve clients. A Servlet or JSP in the presentation tier intercepts client requests, manages logons, sessions, accesses the business services, and finally constructs a response, which gets delivered to client.

**Business tier** provides the business services. This tier contains the business logic and the business data. All the business logic is centralized into this tier as opposed to 2-tier systems where the business logic is scattered between the front end and the backend. The benefit of having a centralized business tier is that same business logic can support different types of clients like browser, WAP, other stand-alone applications etc.

**Integration tier** is responsible for communicating with external resources such as databases, legacy systems, ERP systems, messaging systems like MQSeries etc. The components in this tier use JDBC, JMS, J2EE Connector Architecture (JCA) and some proprietary middleware to access the resource tier.

**Resource tier** is the external resource such as a database, ERP system, Mainframe system etc responsible for storing the data. This tier is also known as Data Tier or EIS (Enterprise Information System) Tier.

**The advantages of a 3-tiered or n-tiered application:** 3-tier or multi-tier architectures force separation among presentation logic, business logic and database logic.

**key benefits:**

• **Manageability:** Each tier can be monitored, tuned and upgraded independently and different people can have clearly defined responsibilities.

• **Scalability**: More hardware can be added and allows clustering (i.e. horizontal scaling).

• **Maintainability**: Changes and upgrades can be performed without affecting other components.

• **Availability**: Clustering and load balancing can provide availability.

• **Extensibility**: Additional features can be easily added.

* **Introduction to J2EE APIs**
* **Java servlet** 
  + The java servlets provide object oriented abstractions for building dynamic web application.
  + Servlets are server side component that are a platform independent high performance replacement for CGI.
  + A servlets class extends the capabilities of servers that host the applications accessed by way of a request response programming model.
  + Although, the servlet can respond to any type of request. They are commonly used to extend the application hosted by the web servers.
  + In short, servlets dynamically generate HTML as a result of a http request.
* **JSP** 
  + The Java Server Pages are an extension of servlet API. It simplifies the generation of dynamic web pages by providing template driven web application development.
  + A JSP page is a text based document that contains **2** **types** of text :

**1**.Static template data which can be expressed in any text based format such as, HTML,WML and XML.

**2**.JSP elements that determine how the page constructs the dynamic content.

* **EJB**
  + Enterprise Java Beans is a component based architecture for developing, deploying and managing reliable enterprise application in production environment.
  + An EJB is a server side body of code with fields and methods to implement modules of business logic. An EJB can be considered as a building block that can be used alone or with other enterprise beans to execute business logic on the J2EE server. Adhering to the enterprise Java Bean Architecture. There are 3 main types of EJB :
    - 1.session beans
    - 2.entity beans
    - 3.message driven beans
* **JDBC**
  + Java Database Connectivity provides platform and vendor independent access to SQL compliant database.
  + Basic JDBC is J2EE.the J2EE platform requires besides a basic APIs also the JDBC 2.0 optional package. JDBC allows transactional querying, retrieval and manipulation of data in JDBC complaint database.
  + Almost, every relational database can be used on the J2EE platform for that needs to use either native JDBC drivers or JDBC- to –ODBC Bridge. It can possible to use the JDBC API from servlet or JSP page to access the database directly without going through an enterprise bean
* **JMS**
  + Java Message Service provides java API for message queuing and publish and subscribe types of message oriented middleware [MOM].It enables distributed communication that is reliable and a-synchronous.
* **Java Mail**
  + This API provides a platform independent and protocol independent frame work to build java based email application.
  + Although platform independent, Java mail depends on underlying email server like send mail or MS Exchange server for the actual sending of on E-mail. It uses IMAP4,POP3[Post office Protocol] and SMTP[Simple Mail Transfer Protocol] as its underlying transport mechanism. Java mail API has two parts : • Application level interface used by the application components to send mail.
  + A service provider interface.
* **JAF**
  + Javabeans Activation Framework is required for java mail. Java mail uses JAF to determine the contents of Multiple Internet Main Extension[MIME] message and what appropriate actions can be done accordingly.
* **JNDI**
  + Java Naming and Directory Interface provides naming and directory functionality.
  + It provides a applications with the methods for performing standard directory operations such as, associating attributes with the objects and searching for objects by using their attributes. Using JNDI, J2EE application can store and retrieve any type of a named java object .
  + J2SE also specifies a JNDI Service Provider Interface [SPI] for naming and directory service providers to implement. JNDI is not a J2EE API but, a J2SE API.
* **JSF**
  + Java Server faces is a relatively new technology that attempts to provide a robust rich user interface for web application. This is used in conjunction with servlets and JSP.
  + JSF provides a component based API for building user interface. These components in JSF are user interface components that can be easily put together to create a server side user interface.
  + The JSF technology also makes it easy to connect to the user interface components to application data sources and connect client generated events to event handler on the server.
* **what is an enterprise? Explain enterprise application**

an enterprise is an business organization.

**Enterprise application :-**

* Enterprise applications are those software applications that facilitate various activity of an enterprise.
* Enterprise application can facilitate the end users through the internet or VPN.(virtual Private Network)
* Well-designed enterprise applications should meet the following goals:

1. Robustness
2. Scalability and performance
3. object oriented design
4. avoid complexity
5. Maintainable and extensible
6. Just-on-time
7. Easy to test
8. Reusable
9. Support for multiple client types
10. Portability

* **Robustness**

Enterprise software is important to an organization and its users expect it to be reliable and bug free as a lot depends on that.hence,we must understand and take advantages of those of parts of J2EE.it helps to build robust solutions and provide a way to write high-quality code.

* **Scalability and performance**

Enterprise application must meet the performance expectation of their users. They must also exhibit sufficient scalability for application to support increased load given to the appropriate hardware.

* **object oriented design**

Object oriented principle offer proven benifites for complex systems.

* **avoid complexity**

Practitioners of extreme programming advocate doing the thing that could possible work.due to the range of components on offer the designer should be able to do design J2EE solutions.

* **Maintainable and extensible**

It is important to consider maintainability while designing J2EE fields.

* **Just-on-time**

The productivity is vital consideration.

* **Easy to test**

Testing is an essential activity throughout the software life-cycle.

* **Reusable**

Enterprise software must feet into an organization long term strategy. thus, it is important to promote reuse. so that, code duplication is minimized.

* **Support for multiple client types**

J2EE applications always and to support multiple J2EE technology client types such as web application ,java applets and so on.

* **Portability**

The purpose of J2EE is to simplify technical complexity for building enterprise application.

J2EE specifies following building for enterprise application.

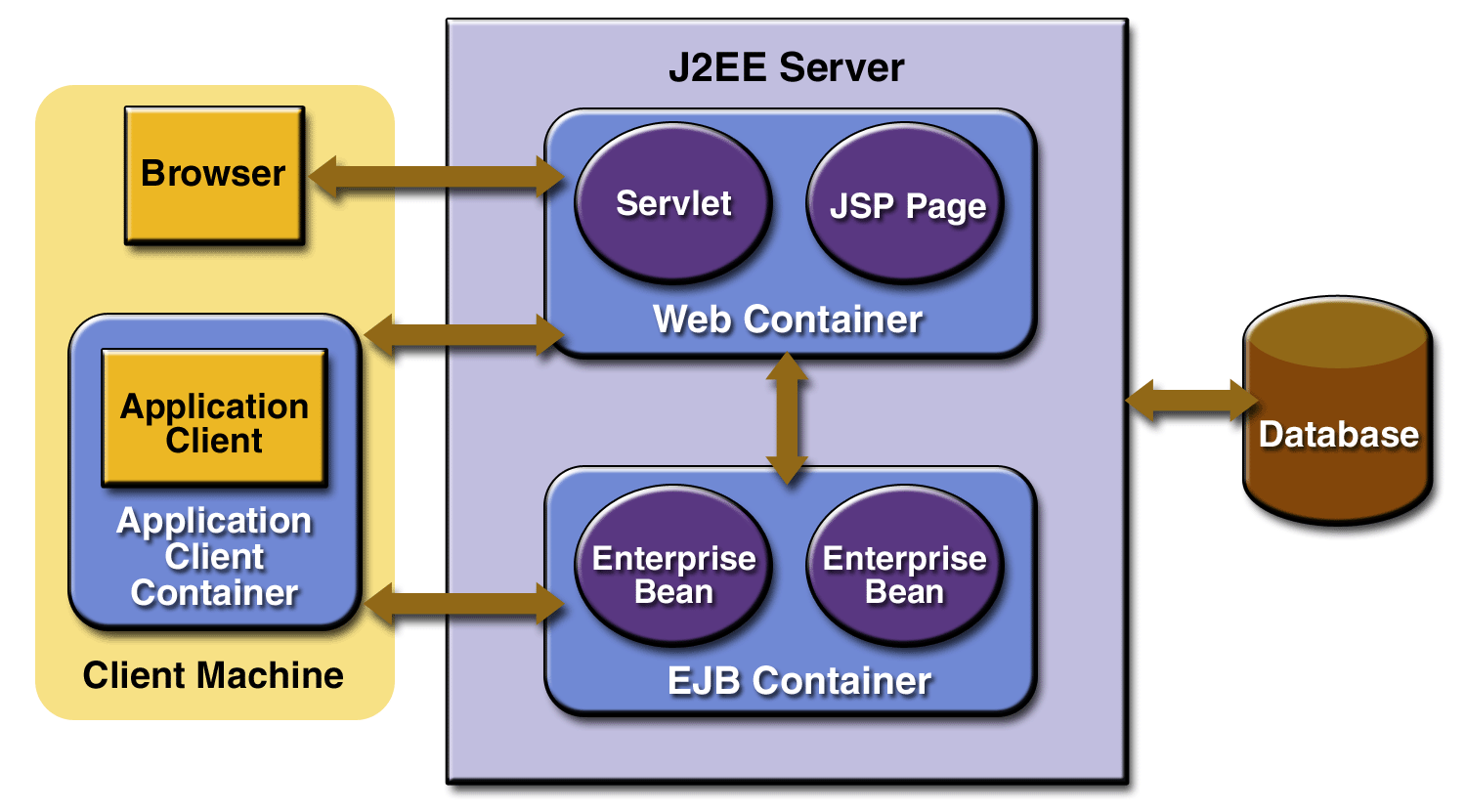
- A programmer model that support set of API.

- Application infrastructure.

* **Introduction to containers**
* There are two types of clients : 1)Thin client 2) Thick client.
* Normally, thin-client multi tiered applications are hard to write because they involve many lines of complex code to handle transaction and state management, multithreading, resource pooling, and other complex low-level details.
* The component-based and platform-independent J2EEarchitecture makes J2EE applications easy to write because business logic is organized into reusable components. In addition, the J2EE server provides underlying services in the form of a container for every component type. There is no need to develop these services, concentrate on solving the business problem at hand.
* For all these components server provides services in the form of container. Because of these container there is no need to write services for each component included in application.
* platform-specific functionality that supports the component. Container provides communication platform between client and component. Before a Web, enterprise bean, or application client component can be executed, it must be assembled into a J2EE application and deployed into its container.

### Container Types

The deployment process installs J2EE application components in the J2EE containers.



**J2EE server**

The runtime portion of a J2EE product. A J2EE server provides EJB and Web containers.

**Enterprise JavaBeans (EJB) container**

Manages the execution of JSP page and servlet components for J2EE applications. Web components and their container run on the J2EE server.

**Web container**

Manages the execution of JSP page and servlet components for J2EE applications.

Web components and their container run on the J2EE server.

**Application client container**

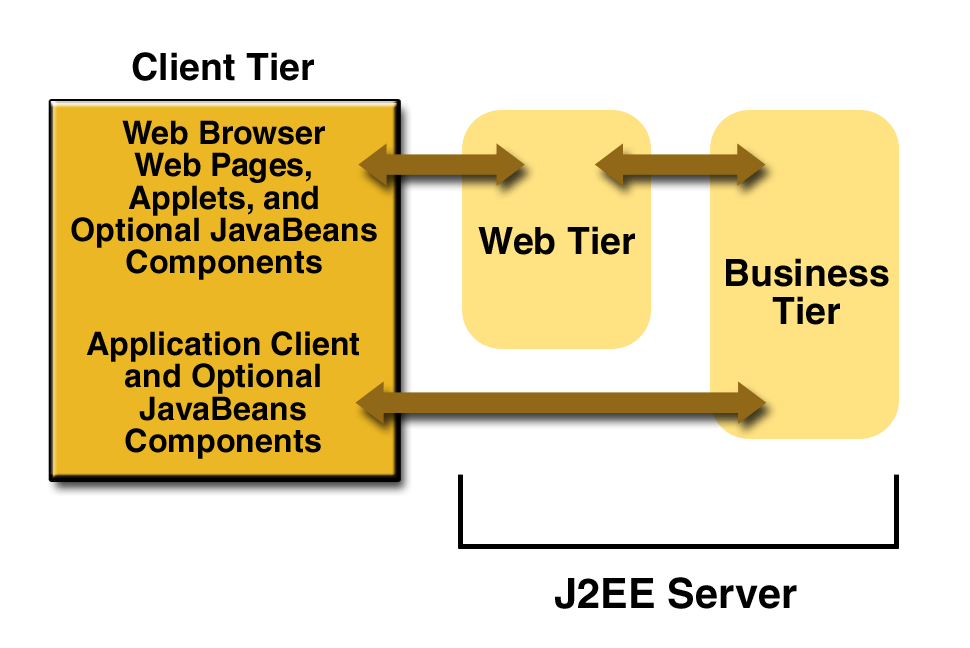
Manages the execution of application client components. Application clients and their container run on the client.

**Applet container**

Manages the execution of applets. Consists of a Web browser and Java Plug-in running on the client together.

#### J2EE Server Communications

* [Figure](http://www.hi.is/pub/cs/2002-03/hv1/j2eetutorial/doc/Overview2.html" \l "79047) shows the various elements that can make up the client tier.
* The client communicates with the business tier running on the J2EE server either directly or, as in the case of a client running in a browser, by going through JSP pages or servlets running in the Web tier.



* J2EE application uses two types of client:
  + A thin browser-based client.
  + Thick application client.
* In deciding which one to use, you should be aware of the trade-offs between

keeping functionality on the client and close to the user (thick client) and off-loading

as much functionality as possible to the server (thin client).

* The more functionality you off-load to the server, the easier it is to distribute,

deploy, and manage the application; however, keeping more functionality on the

client can make for a better perceived user experience.

### Web Components (Web tier)

* J2EE Web components can be either servlets or JSP pages.
* *Servlets* are Java programming language classes that dynamically process requests

and construct responses. *JSP pages* are text-based documents that execute as servlets

but allow a more natural approach to creating static content.

* Static HTML pages and applets are bundled with Web components during application

assembly, but are not considered Web components by the J2EE specification.

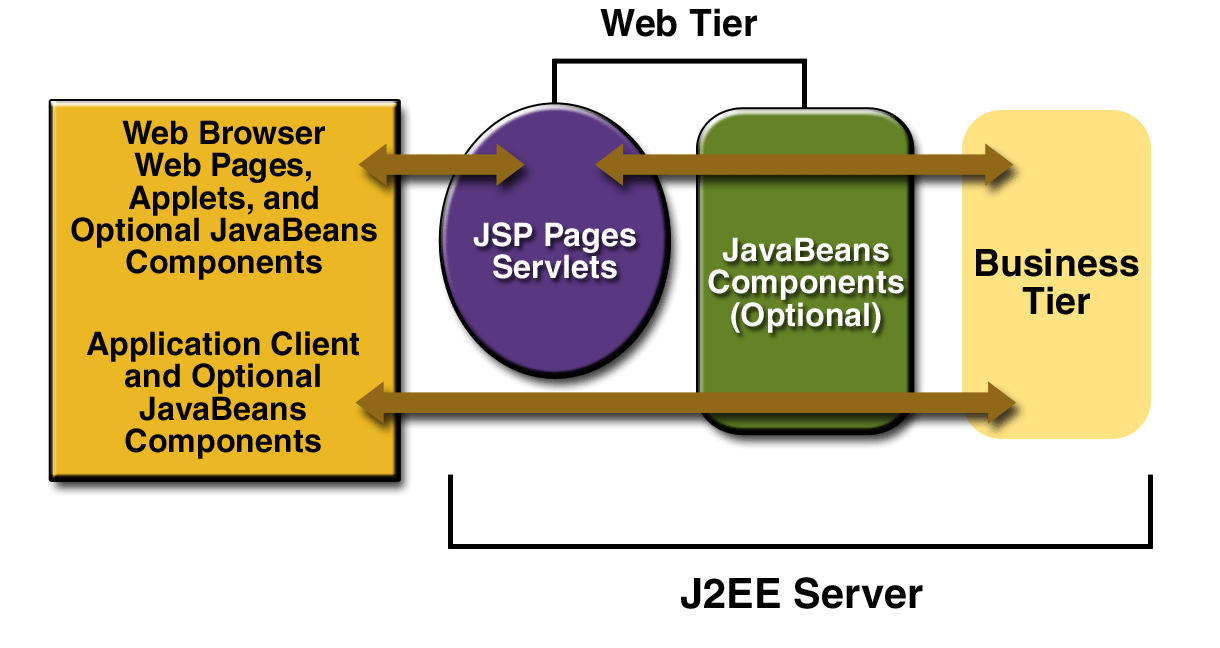
Server-side utility classes can also be bundled with Web components and, like HTML

pages, are not considered Web components.

* Like the client tier and as shown in [Figure](http://www.hi.is/pub/cs/2002-03/hv1/j2eetutorial/doc/Overview2.html#78856) ,the Web tier might include a JavaBeans

component to manage the user input and send that input to enterprise beans running in

the business tier for processing.



### Business Components

* Business code, which is logic that solves or meets the needs of a particular

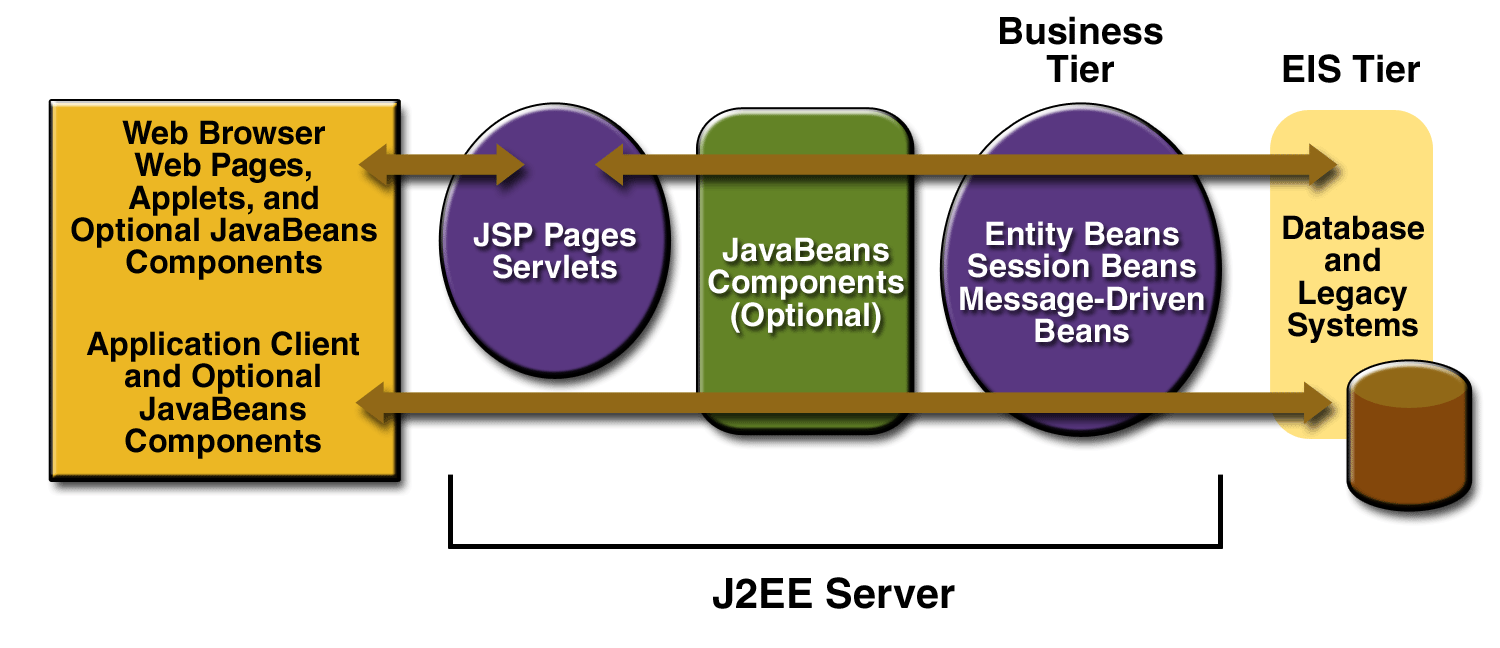
business domain such as banking, retail, or finance, is handled by enterprise

beans running in the business tier.

* Following figure shows how an enterprise bean receives data from client programs, processes it (if necessary), and sends it to the enterprise information system tier

for storage. An enterprise bean also retrieves data from storage, processes it

(if necessary), and sends it back to the client program.



There are three kinds of enterprise beans: session beans, entity beans, and message-driven beans.

* A **session bean** represents a transient conversation with a client. When the client finishes executing, the session bean and its data are gone.
* In contrast, an **entity bean** represents persistent data stored in one row of a database table. If the client terminates or if the server shuts down, the underlying services ensure that the entity bean data is saved.
* A **message-driven** *bean* combines features of a session bean and a Java Message Service ("JMS") message listener, allowing a business component to receive JMS messages asynchronously.

**Enterprise Information System Tier**

* The enterprise information system tier handles enterprise information system software and includes enterprise infrastructure systems such as **enterprise resource planning (ERP)**, mainframe transaction processing, database systems, and other legacy information systems.
* J2EE application components might need access to enterprise information systems for database connectivity.