Chapter 1

Introducing Distributed Applications

Traditional enterprise applications are mostly monolithic applications that integrate all the services offered by them. This leads to the limitation that whenever a new feature has to be implemented, the whole application has to be rebuilt. In addition, these applications have other limitations such as data sharing is not possible and the programming logics involved cannot be reused. These limitations led to the development of distributed applications.

This chapter introduces the distributed application architecture. It also discusses the various technologies used in distributed applications.

Objectives

In this chapter, you will learn to:

- Identify the distributed application architecture
- Identify the distributed application technologies

Identifying the Distributed Application Architecture

Consider a scenario wherein a retail store uses a single-tier computer application to manage sales. The retail store has a single billing counter that uses the application to generate bills. With the increase in the customer base, the owner of the store has expanded the store and started separate sections for each category of items, such as garments, gift items, and general items. These sections are located at different floors.

As there is a single billing counter in the store, all the customers have to come to this counter to make payments. This makes the counter crowded and the customers have to wait for a long time to make the payments. In addition, this increases the workload of the salesperson at the billing counter.

To overcome these problems, the owner of the store has implemented separate billing counters at each floor. Each billing counter has a copy of the computer application to generate bills. This arrangement has solved the problems of the customers and the salespersons at the billing counter. However, it has the following limitations:

- All the billing counters have separate copies of the application that use separate databases. Therefore, all the data in these databases needs to be merged at the end of the day, which is a complex and time-consuming process.
- Being separate applications at each billing counter, it is difficult to manage unique bill numbers.

These limitations can be overcome if all the systems at the billing counters are connected through a Local Area Network (LAN) and the same application can be shared by all the systems. However, the application cannot be shared because it is based on the single-tier architecture.

The single-tier architecture does not support multiple users and is not flexible. Therefore, a single-tier architecture application cannot be distributed over a network. These limitations of the single-tier architecture led to the development of the distributed application architecture.

The distributed application architecture allows multiple users to access an application simultaneously over a network. In addition, each component of a distributed application functions separately and independently of the other components. This increases the performance and scalability of the application.

The Distributed Application Architecture

The application architecture involves breaking up an application into logically connected chunks of code. These chunks of code are called tiers or layers. The application architecture involves the following layers:

- **Presentation layer**: It is the top-most layer in the architecture. This layer provides a user-interface that allows the users to interact and communicate with the applications that further communicates with the server.
- **Business logic layer**: It is the middle layer in the architecture. This layer helps make logical decisions and perform calculations, data updates, and searches that are used in the business services such as user searches. This layer contains algorithms that handle information exchange between the presentation layer and the database server. This layer enables code-reusability by separating the business logic from the presentation layer. In addition, any modification in the business logic does not need the data access layer or the presentation layer to be recompiled.
- **Data access layer**: It is the bottom-most layer of the application architecture. It enables you to access the data stored in various data sources such as RDBMS, XML, and text files. The data access layer enables a developer to use the same layer for accessing data from any database without modifying the data access code. In addition, any modification in the data access code does not need the business logic layer or the presentation layer to be recompiled.

These layers may be placed on different or on the same computer depending upon the business requirements. In the single-tier architecture, these layers are packaged on the same computer. Being placed on the same computer, these layers are tightly coupled. This means that if you make changes to any one of them, you also need to modify others.

In the distributed application architecture, the presentation layer, business logic layer, and the data access layer are placed on different computers to enable reusability and data sharing. Depending upon the location of these layers and business requirements, the distributed application architecture is categorized in the following types:

- Two-tier architecture
- Three-tier/n-tier architecture
- Service-oriented architecture

Two-Tier Architecture

In the two-tier architecture, the presentation layer resides on the client, the business logic layer may reside either on the client or on the server, and the data access layer resides on the server. Depending on the business requirements, an organization can have the following types of two-tier application architecture:

- A fat client and thin server application architecture
- A fat server and thin client application architecture

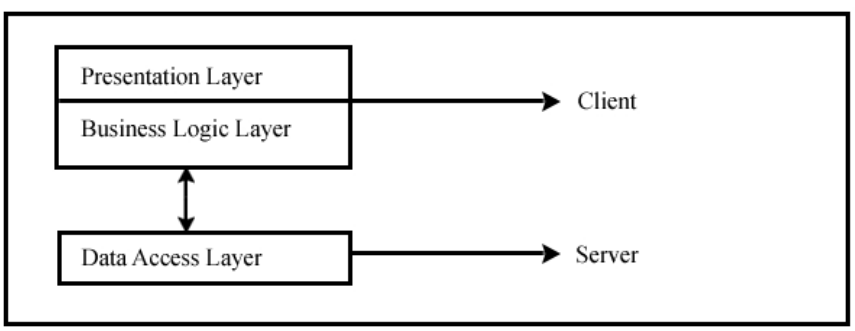
A Fat Client and Thin Server Application Architecture

In this type of application architecture, the presentation layer and the business logic layer reside on the client, and the data access layer resides on the server. Such an application architecture helps in reducing load on the server because the server only stores the data and does not participate in business logic processing, which is done on the client.

This type of application architecture is generally used in organizations that cannot afford a reliable network infrastructure or have less user base.

For example, this type of architecture can be implemented in a Point of Sales (POS) application for retail stores with limited users. In the POS application, the client computers contain the interface that helps enter details related to the sales transactions. In addition to the interface, the client computer contains the logics for bill creation and data validation. However, the server is only used to store or retrieve the data entered or requested by the application at the client computer periodically.

The following figure displays the layout of the layers in a fat client and thin server application architecture.



A Fat Client and Thin Server Application Architecture

A Fat Server and Thin Client Application Architecture

In this type of application architecture, the presentation layer resides on the client, and the business logic layer and the data access layer reside on the server. Such application architecture is easily managed, less prone to security risks, and offers low maintenance and licensing costs. This is because it is easy to manage a single server instead of managing each client computer. Similarly, securing and maintaining a single computer (server) is easier than managing several computers (clients).

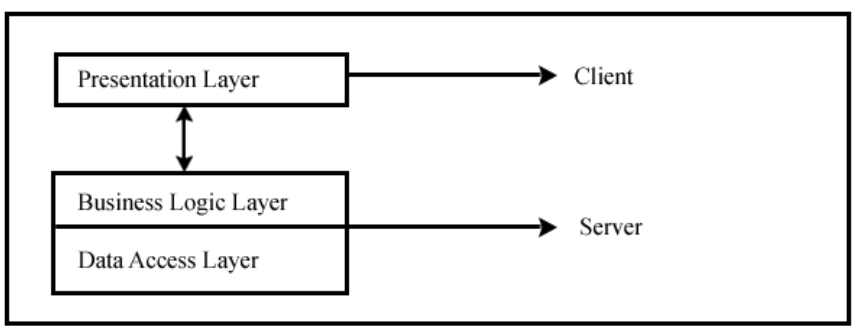
This type of application architecture is primarily used in organizations where the business and managerial operations that need to be performed are complex and large in number and the size of the workgroup that requires access to the server is higher in number.

Consider an organization that wants to implement a computer system that enables the sales managers to generate reports based on the historical sales data. These reports are required to develop sales strategies.

The management at the organization does not want to invest a huge amount in developing the system or in purchasing the hardware.

This is an ideal situation for implementing a fat server and thin client application architecture. This is because in a fat server and thin client application architecture, all the business logic resides on the server. Therefore, the processing is done on the server. This saves cost because only the server needs to have a high hardware configuration. The clients do not require high configuration and can even be dumb computers. In addition, this architecture is less prone to security risks because the security is required only at the server.

The following figure displays the layout of the layers in a fat server and thin client application architecture.



A Fat Server and Thin Client Application Architecture

The two-tier architecture is used for distributed computing with the server and the clients placed at different locations. However, with an increase in the size of the workgroup, the performance of the application may deteriorate in this architecture. This is because the

server may become overloaded when there is a substantial increase in the number of requests it receives. To achieve scalability and robustness, the three-tier/n-tier architecture can be used to design the applications.

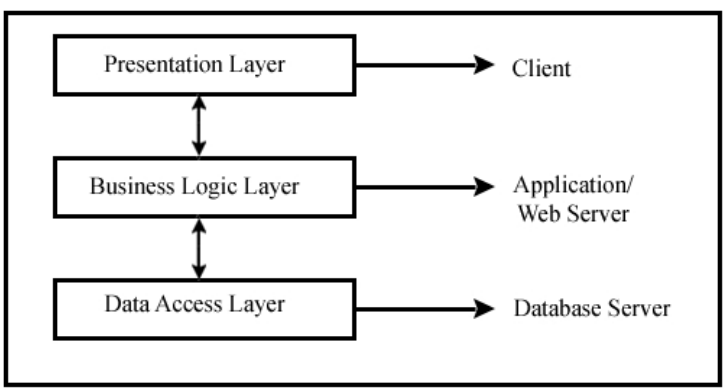
Three-Tier/N-Tier Architecture

In the three-tier architecture, the business logic layer is separated from the data access layer and handled by a middleware. The middleware is implemented by using application servers or Web servers. The middleware performs functions such as queuing of tasks, scheduling, and prioritizing of business processes.

In the three-tier architecture, every tier has its own set of processing power to service client requests. This helps in distributing the user request load resulting in high performance and scalability.

Consider an online shopping application. It is based on the three-tier architecture. In such an application, the presentation layer provides the interface for services such as browsing merchandise, purchasing, and updating shopping cart contents. The business logic layer calculates the total amount including the delivery charges and sales tax that the user has to pay and validates the credit card. The data access layer contains the logic that is used to store and retrieve data in the database.

The following figure displays the placement of layers in the three-tier architecture.

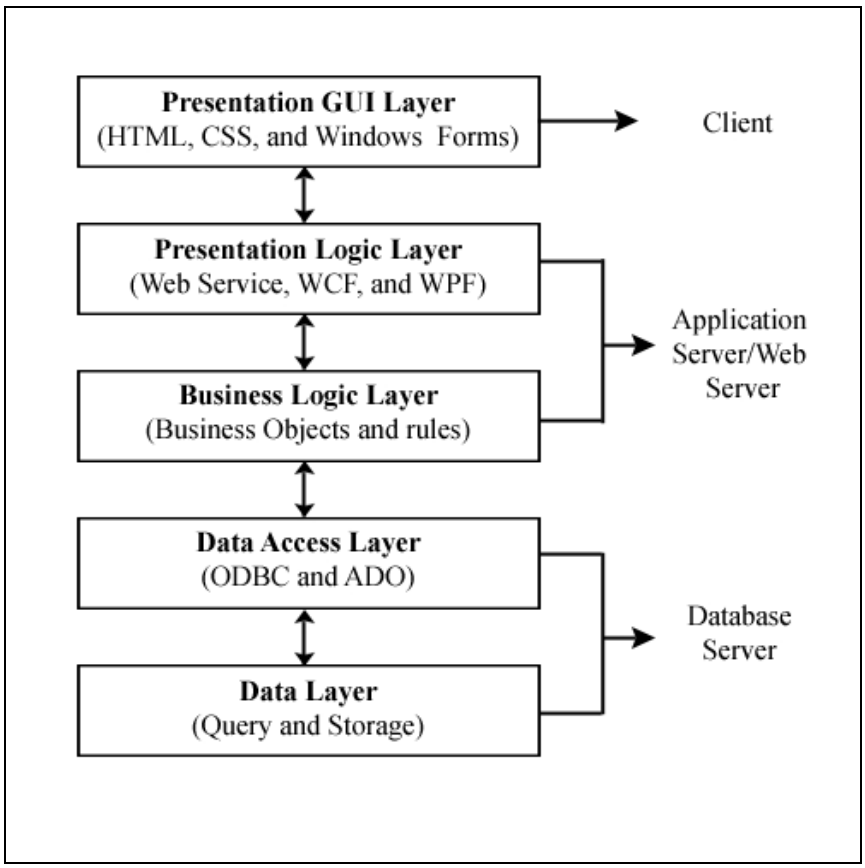


The Three-Tier Architecture

The three-tier architecture is implemented when an effective distributed client/server architecture that provides increased performance and scalability is needed. However, designing and deploying applications based on three-tier architecture is a tedious task. The separation of presentation, business logic, and database access layers is not always obvious because some business logic needs to appear on all the layers. Therefore, there is a need to further divide these layers. This requirement has led to the expansion of the three-tier application architecture into the n-tier application architecture.

A common approach to implement the n-tier architecture involves further separating the presentation layer and the data access layer. The presentation layer is divided into presentation GUI and presentation logic layer, and the data access layer is divided into the data access layer and the data layer.

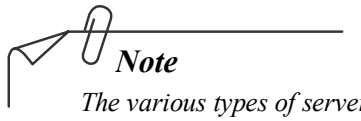
The following figure displays the placement and grouping of various layers in the n-tier architecture.



The N-Tier Architecture

The n-tier architecture separates the various business processes into discrete units of functionality called services. Each service implements a set of related business rules that define what must be done and how it must be done within an organization.

The business rules used in the n-tier application architecture are implemented at various servers, such as application server, database server, and Web server. As a result, the n-tier application architecture provides improved flexibility and scalability of applications as compared to the three-tier application architecture. This is because the functioning of each layer is completely hidden from the other layers making it possible to change or update one layer without recompiling or modifying the other layers.



Note

The various types of servers are:

- ***Application Server**: An application server consists of the business logic in a distributed application and provides services that enable a client application to communicate with the database server.*
- ***Database Server**: A database server is a server where the database resides. The database server provides various database services such as enabling a Web application to update data in the database.*
- ***Web Server**: A Web server is a server that delivers Web content such as Web pages to Web browsers.*

The applications deployed by using the n-tier architecture can be invoked over the Web. Today, most enterprise-distributed applications use Web services that if deployed once on a server can be used by other applications. Web services are platform-independent interfaces that help communicate with other applications developed by using HTTP and XML.

With the advent of various languages and protocols, such as Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP), for sending service requests, Web services can now share data and functionality when using different technologies. Enterprise distributed applications that use Web services are implemented and developed using the Service-Oriented Architecture.

Service-Oriented Architecture

Consider a scenario wherein an organization, ITtech, is focusing towards acquisition and merger activities with another organization, SoftTech. For this, ITtech has to reorganize its internal structure and adopt the technologies and platforms being used by SoftTech. At the same time, ITtech cannot discard its existing business processes, applications, data, and other information systems because they are reliable and stable, and a sizeable investment has been done on them. Therefore, ITtech needs to adopt a strategy that will enable it to integrate the existing applications and business processes with the applications and business processes used by SoftTech. This can be achieved by implementing SOA. This is because SOA provides interoperability and cross-platform communication.

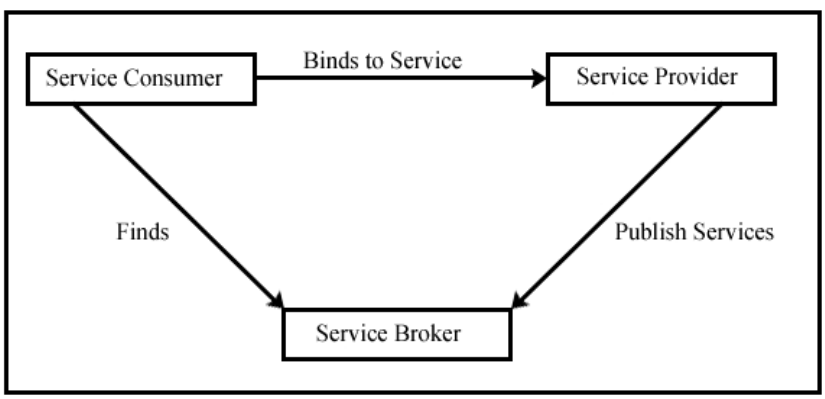
SOA is an n-tier application architecture that is used to develop loosely-coupled distributed applications. This architecture allows various applications built on the same or different platforms to communicate with each other over a network because it makes use of standard technologies and protocols. With SOA, organizations can retain their legacy software system and acquire the benefits of the latest technologies and interoperability in a cost-effective way.

SOA consists of smaller modules of a distributed application known as services. A service is a well-defined and self-contained function that does not depend on the state of other services. These services are exposed as messages that can be accessed over the Web. Such services are known as Web services. SOA is implemented by using Web services. Web services are the components that consist of business logic and are language and platform independent.

The SOA architecture consists of the following building blocks:

- **Service provider**: Publishes a service and registers it in Universal Description, Discovery and Integration (*UDDI*), which is a public registry that contains the address and description of a service. Registering a service in the UDDI enables various service consumers to access the service.
- **Service broker**: Enables the service consumers to find a service provider that offers the service required by the consumer. Once the service consumer finds the appropriate service, the service broker binds the service consumer to the service.
- **Service consumer**: Finds the service according to the requirements through service broker and then accesses the service provided by one or more service providers.

The following figure displays the building blocks of SOA.



The Building Blocks of SOA

The SOA architecture allows client applications built on different platforms to access the service over the Web. This is because SOA uses Simple Object Access Protocol (SOAP) to exchange messages between a Web service and a client application.

SOAP is a lightweight XML-based messaging protocol used to convert the data into SOAP messages before sending them over a network. SOAP provides interoperability and extensibility in a distributed application, thereby, enhancing the distributed application architecture.

Advantages of SOA

The advantages of implementing SOA in a distributed application are:

- **Offers services across platforms**: SOA allows you to develop services that can be accessed by various client applications built on different platforms. For example, a Web service can be accessed by a java client application and by a .NET client application.
- **Offers location independence**: A service based on SOA can be accessed from any location and at any time. You only require a Web browser and a Web application to access the service.
- **Offers dynamic search and binding of services**: The service broker allows you to search the various services offered by the service provider and binds the client application with the appropriate service.
- **Offers scalability and flexibility**: SOA allows you to create a loosely coupled application in which all the layers of the application functions independent of each other. This increases the scalability and flexibility of the application.

Advantages of the Distributed Application Architecture

The distributed application architecture provides the following advantages:

- **Increased productivity**: Multiple users cannot access single-tier applications simultaneously. This affects the productivity of an organization because if multiple users need to use the application, they have to wait until the application is available. However, this is not the case with distributed applications. Multiple users can access the application simultaneously. This increases the productivity of an organization.

  Similarly, distributed applications increase the productivity of the developers. This is because the functionality of distributed applications is divided into separate layers/tiers and these layers/tiers are independent of each other. Therefore, different developers can develop these layers/tiers. This saves time and increases the overall productivity because the developers' tasks are not interdependent.
- **Flexibility and scalability**: The functionality of distributed applications is divided into independent layers/tiers and these layers reside on different computers. This provides flexibility to distributed applications because it allows developers to modify the functionality of a layer without affecting the other layers. In addition, it provides scalability because each layer resides on a different computer and functions independently, allowing the application to handle and process multiple users' requests without affecting the performance.

  The flexibility feature enables an enterprise to enhance the functionality of a layer without affecting the other layers. Similarly, the scalability feature enables an enterprise to increase the numbers of employees accessing the distributed application without affecting the performance of the application.

- **Fault tolerance and availability**: In the single-tier application architecture, all the layers reside on the same computer. If the computer crashes or there is a power cut-off, the application ends resulting into data loss and inconsistent data. In addition, it delays the user's work because the user has to restart the application in order to continue working.

  In the distributed application architecture, all the layers are independent of each other and reside on separate computers. If any of the computers crashes or becomes unavailable due to network problems, only the functionality that resides on that computer becomes unavailable. The rest of the functionality remains available to the end-users. This feature benefits an enterprise because if a client application fails to work properly on an employee's computer, only that employee will get affected. Other employees can carry on with their work.
- **Reusability**: In the distributed application architecture, the developers create the business logic as a reusable piece of code known as a component. A component is loosely coupled and can be reused when similar functionality is required in any other application. This saves the time and effort of a developer because the same code need not be rewritten when similar functionality is required in another application.

Identifying the Distributed Application Technologies

When developing a distributed application, you need to decide on the technology that is to be used for the application. The choice of the technology depends upon the application requirements. There are various technologies that can be used to develop a distributed application in .NET. Some of the technologies are:

- COM+ services
- .NET Remoting
- Web services

COM+ Services

Single-tier applications provide code reusability with the help of components. *Components* are reusable piece of codes that contain the business logic of an application and are developed by using Component Object Model (COM). However, these components have a limitation that they cannot communicate with other applications in a distributed environment. This led to the development of Distributed Component Object Model (DCOM).

DCOM is used to develop a component that can communicate with remote applications across a network. The DCOM programming model became the base for all the distributed applications. However, the distributed applications based on the DCOM programming model also have a few limitations. To understand these limitations, consider a scenario of a retail store having different sections for apparels, cosmetics, and gift items. These sections have separate billing counters that use a distributed application to store the details of the sales transactions in a centralized database and to generate bills. This application uses a component to interact with the centralized database.

With the increase in the number of customers, the number of billing counters has increased. As a result, multiple sales persons at different billing counters access the component simultaneously to interact with the database. This leads to the poor performance of the server, where the component is residing. In addition, multiple clients accessing the component results in network congestion. This leads to unreliability of data transfer and inconsistency in data.

These problems can be solved by implementing COM+. *COM+* is an object model used to develop components in a distributed application environment. COM+ provides features such as reliable data transfer, transaction management, sharing of resources, and optimized consumption of memory that help developing robust components and provide easy communication between distributed applications. These features are called COM+ services.

.NET Remoting

There may be situations where a component needs to be accessed over an intranet as well as over the Internet by client applications that are based on similar platforms. In such situations, more than one protocol is used to provide communication between the client applications and the component. For example, client applications use Transmission Control Protocol (TCP) to access the component over an intranet and Hypertext Transfer Protocol (HTTP) to access the component over the Internet.

Consider the scenario of a bank that has various branches at different locations in the U.S. and head office in California. The bank employees use a distributed application to manage their daily work. In the distributed application, the database server and the application server reside at the head office. The application server contains a component that is created by using the .NET platform.

The employees at all the branches and the head office use a .NET client application to interact with the database server. The client application interacts with the component and the component further interacts with the database server to enable communication between the client application and the database server.

Being at the same location, the employees at the head office access the application server through LAN whereas the employees from other branches access the application server through the Internet. As both the networks use different protocols, the client application and the component need to implement .NET Remoting to enable communication through different protocols.

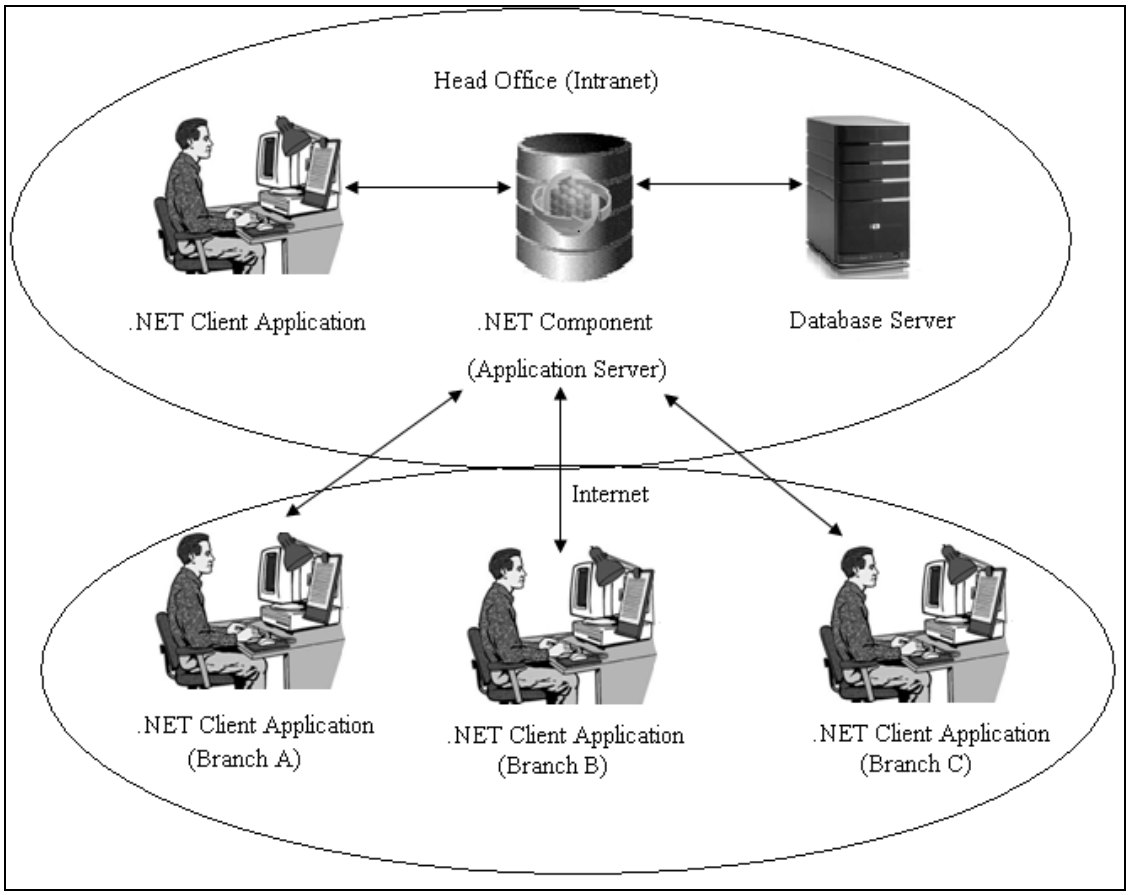
.NET Remoting enables a .NET application and a .NET component residing on remote computers to communicate with each other using various protocols, such as TCP and HTTP. This makes the communication easy and flexible.

.NET Remoting will enable the bank to allow the employees at the head office to access the component through LAN by using the TCP protocol. The TCP protocol uses a binary format (proprietary .NET format) to transfer data between the component and the client application providing better performance.

Similarly, .NET Remoting will enable the employees from other branches to access the component through the Internet by using the HTTP protocol. The HTTP protocol does not require a continuous connection providing the optimum use of the bandwidth.

In this way, the requirement for both type of clients (local and remote) can be handled by implementing a single technology.

The following figure illustrates the scenario of the bank.



Communication Between the Client Applications and the .NET Component

Note

To implement .NET Remoting, you need to create the client application and the component by using .NET.

Web Services

There may be a situation where you want your application to be accessed by client applications irrespective of the language and platforms the client applications are using.

Consider a scenario wherein you need to develop a server application that provides latest information on share market. You want that the server application is accessible over the Web. For this, the server application should be interoperable because being on the Web it must be accessible to any client application irrespective of the language or platform the client application is using. Such an application can be created by using Web services.

A *Web service* is a component that implements program logic and provides functionality that can be accessed by disparate client applications over the Web. Web services enable heterogeneous applications to interoperate with each other. This is because Web services use XML-based messaging to send and receive data and XML is a standard that is understandable by all systems.

Practice Questions

1. Which of the following technologies enables a client application to communicate with a component by using protocols such as TCP and HTTP?
   a. COM
   b. DCOM
   c. .NET Remoting
   d. Web services

2. In which type of application architecture, the data access layer is divided into data access layer and data layer.
   a. Two-tier architecture
   b. N-tier architecture
   c. Single-tier architecture
   d. Three-tier architecture

3. Which of the following technologies allow an application to access a component over the Internet?
   a. COM+ services and .NET Remoting
   b. Web service and .NET Remoting
   c. COM+ services and Web services
   d. DCOM and COM+ services

4. Which of the following distributed application architectures is used to develop a Web service?
   a. Three-tier architecture
   b. N-tier architecture
   c. Two-tier architecture
   d. Service-oriented architecture

5. Which of the following layers enables a developer to access data from the databases?
   a. Presentation layer
   b. Data access layer
   c. Business logic layer
   d. Presentation GUI layer

Summary

In this chapter, you learned that:

- The layers involved in the application architecture are:
  - Presentation layer
  - Business logic layer
  - Data access layer
- The various types of distributed application architecture are:
  - Two-tier architecture
  - Three-tier/n-tier architecture
  - Service-oriented architecture
- The advantages of distributed application architecture are:
  - Increased productivity
  - Flexibility and scalability
  - Fault tolerance and availability
  - Reusability
- The various types of technologies that can be used to develop a distributed application in .NET are:
  - COM+ services
  - .NET Remoting
  - Web services