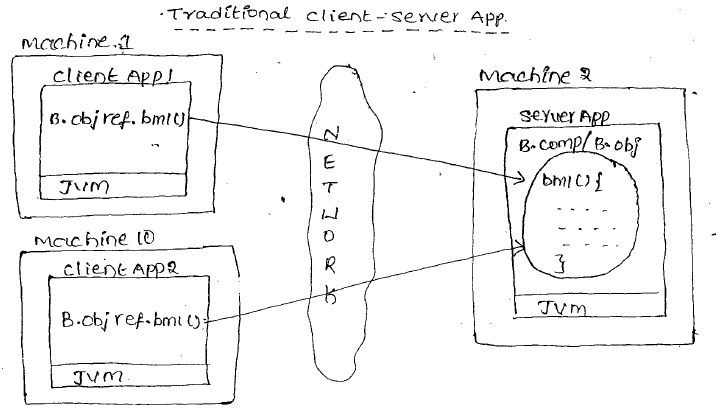
**Java Web Services**

**Introduction:**

Web Service is the technology that allows us to build **interoperable distributed applications**.

**Understanding traditional client-server application:**

In traditional **client-server application** the client application interacts with the business object of server application directly. So, if there is any change in the location of server application we must modify the code of the client application to interact with the server application that is changed to new location. This is called “**Location Dependency”.**



Socket programming based client-server applications are 2-tier application/ traditional client-server applications.

**RPC (Remote Procedure Call):**

Remote Procedure Call (RPC) is a [protocol](http://searchnetworking.techtarget.com/definition/protocol) that one program can use to request a service from a program located in another computer on a [network](http://searchnetworking.techtarget.com/definition/network) without having to understand the network's details. A procedure call is also sometimes known as a **function call** or a **subroutine call**.

* Applications must be developed in C, C++ technologies.
* Platform dependent (OS Dependent).
* Architecture dependent.
* Language dependent. (Both client and server applications must be developed in same language i.e. C and C++).
* Outdated Technology

**Drawbacks of Traditional Client-Server Applications:**

1. These applications are location dependent.
2. Keeping multiple B- components in one server application increases burden on the server application. But keeping multiple B- components in multiple server applications makes each client talking with multiple server applications having location dependency.

To solve the above problem we work with distributed technologies.

**Distributed Programming**

**Layman Terms:**   
  
We have a piece of work or Job or Task, if it is assigned to one person he needs 100 hours of time to build the solution. But I want this job to be completed much lesser time than this.   
To reduce the amount of time needed, there are multiples ways, out of which few are:  
  
**Trial 1:** **Increase the energy efficiency levels of the resource:**.

When in the morning if the worker comes to shop, I will call him and ask what he has eaten today for breakfast. Let’s assume that he said that he ate 2 idly, I will tell him that you are working this hard, and food you ate is very less and I will offer him 10 rupees daily for breakfast and I will ask him to eat 4 idly. So, indirectly am increasing the energy efficiency levels of a worker. So, worker works with more energy and the 100 hours work will now be reduced to 90 hours. So, am reducing the job completion time of a worker by adding 2 more idly to his breakfast.  
After a week if I repeat the same procedure and now I ask him to eat 6 idly by offering him more money daily, the job completion time will further be reduced from 90 hours to 82 hours.  
  
If I keep on increasing the offering idly every week then also the job completion time will be further reduced from 82 hours to 79 hours and then from 79 hours to 78 hours.. Here you can see nominal change in performance.   
  
But now if we further offer him more idly to eat for breakfast it’s useless. The single worker can do work to the job in maximum of 78 hours, even we keep on increasing the idly count too. Here the performance improvement might be observed but it would not be always relatively impacted. It’s beyond his capacity to further reduce the job completion time.

2 Idly

100 Hours

4 Idly

90 Hours

6 Idly

82 Hours

8 Idly

79 Hours

10 Idly

78 Hours

**Trial 2: Engineer the system/Increase the skill set of a system.**Train the worker by sending him to Ameerpet training centers and ask him to learn some techniques on how to sell the clothes or how to increase the communication techniques. Ask him to implement that techniques in his daily works, this will further reduce the job completion time by some hours.

78 Hours

1 Course

75 Hours

2 Courses

72 Hours

**Trial 3: Distribute job across various people.**

As there is no way to further reduce the work from 72 hours by a single worker, now the owner of the shop decided to complete the same job by hiring one more resource then the work got completed by both the workers by almost 40 hours. Then he employed one more worker and the work further reduced to 30 hours.. and so on.

**Technical Terminology :**

We have a piece of work or Job or Task, if it is assigned to one person he needs

100 hours of time to build the solution. But I want this job to be completed much lesser time than this. To reduce the amount of time needed, there are multiples ways, out of which few are:

1. **Increase the energy efficiency levels of the resource, in other words scale up the hardware of the machine:**

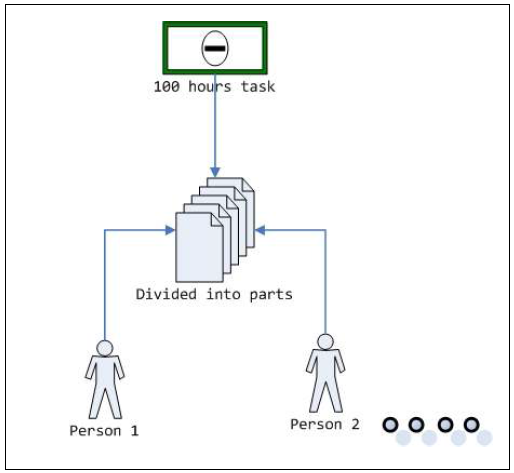
Let’s say if the machine has been configured with 2 GB RAM and takes 100 hours of time to compute the solution, increase the RAM from 2 GB to 4 GB, so that the Job would be computed quicker within 95 hours. Increasing the RAM size from 2 GB to 4 GB has improved the performance and reduced the processing time by 5 hours so, let’s say if I increase the RAM from 4 to 8 or 8 to 16 or 16 to 32, the performance improvement might be observed but it would not be always relatively impacted. You might see performance improvement up to certain extent but beyond that they may not be any further effect in performance even you scale up the system.

1. **Engineer the system:**  
     
   To build a solution for a job always there are multiple ways of doing it, but to complete it within less amount of time we should have a proficient skill set in working on it.

When writing logic, we can produce the same output by writing 10 lines of code and even with 5 lines of code, to reduce the amount of code to build the solution; we need to engineer our system. By modularization and effective design we can always reduce the number of lines and tune the logic to complete it faster. The more you engineer/tune the less the time required to process it. But this would also not always relative, because even overdose of medicine will acts as poison, beyond certain extent if you try to re-use or reduce the amount of lines, rather than giving a positive effect; it might downgrade the performance of the application itself.

By now we understood that we can reduce the turnaround time that is required to execute a job by one person, but that is up to certain limit, may be could be 10% or 20% of the actual time, but we may not surprisingly bring down it to half or quarter of the original time by single person.

But there are use cases that demands high amount of throw put or less processing time, such kind of jobs cannot be handled by one person rather we need to distribute such jobs between more than one people. The same is shown in the below figure.



In the above diagram, it shows a 100 hours of job has been divided into multiple parts (let’s say two equal 50 hours) and each part is independent with other part. The parts are assigned or shared across two or more people, so that each one would be able to execute on its own piece to build a partial solution, as there are working parallel the time required to complete 100 hours of job by two person’s working parallel is 50 hours and the time required to combine their individual outputs to form the final output may be 5 hours. So, the total time required to build the final solution would be 55 hours which when compare with one person working on that job it’s a great performance improvement and drastic change in turnaround time.

So, if you want greater performance improvements or quicker turnaround time such kind of jobs must be distributed across various people.

Initially we have programs which are designed to run on one Box/Machine. Like when we develop “C” language program, it will execute on a machine and can talk to other programs which are running on same machine. In this case if we run two instances of the program “P1” on the same machine both instances will execute simultaneously by sharing the totally hard of the same machine. In this case they won’t be any performance improvement as they cannot scale up beyond the current machine configuration.

Now we are looking to build applications/programs which not only execute on a machine and talk to other programs on the same machine, these programs should also be able to access/communicate with other programs which are running on different machine. In order for a program to talk to other program running on different machines, both the machines should be connected over wire/network. So, we are trying to build applications which can talk over the network, which means network enabled or network aware applications. By now we understood distributed applications means the programs which are designed to talk to other programs which may be running on same or physically separated but connected via network machines as well.

**Advantages of Distributed programming**

There are several advantages of going for distributed programming few of them are as described below.

1. **Higher throughput:**   
     
   In distributed programming the number of jobs being computed within a particular time would be more when compared with one program running on one machine.  
   **Throughput:** Throughput is a measure of how many units of information a system can process in a given amount of time.  
   **Example**:

1 hour 🡪 20 jobs

2 hour 🡪 40 jobs

1. **Quicker turnaround time:**

If a job takes 100 hours of time to complete, if we distribute this job across multiple machines and executed in parallel, it would be completed much faster than 100 hours. So, turnaround time would be faster when we distribute it.

**What is the turnaround time?**

In computing, turnaround time is the total time taken between the submission of a program/process/thread/task (Linux) for execution and the return of the complete output to the customer/user. It may vary for various programming languages depending on the developer of the software or the program.  
 Turnaround time= End Time – Start Time

1. **Highly Responsive:**

The time taken for the program to respond to the inputs must be very fast.

**Layman Example**

If you are about to catch a train at the last moment and the ticket counter took money from you and the ticket is taking more time to get printed, we consider that as less responsive. If the ticket gets printed soon, it’s considered as a highly responsive machine.

1. **High availability of resources:**

If a job has been assigned to one person to execute, if he falls sick they won’t be any other person who is going to back up the first person to complete it. In case if we distribute it across more than one resource and if one falls sick always there is other people/resources that can share the load of the person and can complete it.

1. **High utilization of resources:**

There are some resources which are very costly, let’s say printer. If want every employee in the organization to have access to printer, instead of buying 100 printers for 100 employees, we can have one printer which can shared across 100 employees so that it would be utilized by everyone effectively.

**6. Location Transparency**

If distributed applications registry software must be there in a fixed location. But client and server applications may change their locations. In distributed applications clients talk with server applications only after getting B- Object reference from the registry. So, any change in the location of server application, we just need to inform the registry software and there is no need of informing to client applications and modify their code. This is called **Location Transparency** or **location Independency.**

**7. Reuse already developed(old) functionality into new software:**

Let’s understand with very simple example. Let’s say you are developing finance software for a company on java and you have old .net software which manages salary of employees. So rather than developing new software for employee part, you can use old software and for other parts like infrastructure you can develop your own functionalities.

Web Services allow the business logic of many different systems to be exposed over the Web. This gives your applications the freedom to choose the Web Services that they need. Instead of re-inventing the wheel for each client, you need only include additional application-specific business logic on the client-side. This allows you to develop services and/or client-side code using the languages and tools that you want.

**8. Interoperability:**

This is the most important benefit of Web Services. Web Services typically work outside of private networks, offering developers a non-proprietary route to their solutions. Web Services also let developers use their preferred programming languages. In addition, thanks to the use of standards-based communications methods, Web Services are virtually platform-independent.

9. **Loosely Coupled:**

Each service exists independently of the other services that make up the application. Individual pieces of the application to be modified without impacting unrelated areas.

1. **Ease of Integration:**

Data is isolated between applications creating ’silos’. Web Services act as glue between these and enable easier communications within and across organizations.

**Silo**: a pit or underground space for storing grain, green feeds, etc.

1. **Deployability :**

Web Services are deployed over standard Internet technologies. This makes it possible to deploy Web Services even over the fire wall to servers running on the Internet on the other side of the globe. Also thanks to the use of proven community standards, underlying security (such as SSL) is already built-in.

**What is distributed program?**

**Machine 1**

Here, more time is required to solve the programs, because till the time first program P1 is not completed, the resources are not released to execute P2.

**Machine 2**

**Network**

Here resources are being shared effectively and P1 and P2 are executed in parallel and effective utilization of resources is occurring.

Here, P1 and P2 not only contain business logic, it also contains the network related code to connect with the other program. P1 needs network related code to connect to P2 and vice versa.

**Distributed Technologies**

**The following are the different Distributed Technologies available now**

* RMI (Remote Method Invocation)
* CORBA (Common Object Request Broker Architecture)
* Microsoft DCOM (Distributed Component Object Model)
* Web Services
* SOA Suite
* EJB (Enterprise Java Bean)
* COM, COM+/DCOM
* RPC (Remote Procedure Call)
* JINI

**All Distributed Technologies uses its own protocols like**

* + **RMI** uses **JRMP Protocol**.
    - RMI stands Java Remote Method Protocol   
      New version of RMI uses RMI-IIOP   
      RMI-IIOP stands Java Remote Method Invocation (RMI) interface over the Internet Inter-Orb Protocol.
  + **CORBA** uses **IIOP Protocol**.
    - **IIOP** stands for Internet Inter-Orb Protocol   
      Here **ORB** stands for Object Request Broker.
  + **Microsoft COM/DCOM** uses **DCOM Protocol**.
  + **RPC** uses **PRC protocol**.
    - RPC stands for Remote Procedure Call
  + **Web Services** uses **SOAP protocol**.
    - SOAP stands for Simple Object Access Protocol

Here we have ‘n’ number of distributed technologies. But most of the companies are using Web Services to implement their Distributed Applications.

**Java role in Distributed Programming**

Every programming languages added support to build distributed applications.

Equally Java also has its contribution towards it.

Java has various API’s which allows us to build distributed/remote applications.

1. **CORBA**
2. **RMI**
3. **EJB**
4. **Web Services**

Let’s try to understand closely the API’s in Java, their advantages and disadvantages of each, and what are the factors driving us to Web services.

**CORBA**

CORBA stands for **Common Object Request Broker** **Architecture**. After socket programming, the first API that has release by Java to support Distributed programming is CORBA.

In CORBA programmer has to code using **IDL**.   
  
IDL stands for “**Interface Definition Language**”, which is a **language independent/neutral code**, it is a scripting language where CORBA developer writes the IDL file and **gives it to CORBA compiler**.   
  
The **CORBA compiler** will generate the language specific object (let’s say Java or C or C++ or .net) into the generated Object, i.e. it **gives skeleton representation** in any language, and the programmer has to write the business logic.

After **writing the business logic**, in order to **expose the CORBA** object over the network it has to be deployed on **MOM’s**.

**MOM** stands for **Message Oriented Middleware** and is nothing but a CORBA server. The purpose of MOM is to host and expose only the CORBA objects. But in order to use MOM, we need to acquire license and involves cost (not open source).

Considering all the above factors, like development will starts with IDL (seems to be different from general way of programming) and deploying requires a licensing server, at those times working would CORBA seems to be complicated and quite messy. This makes CORBA quickly vanish from the market.

CORBA uses **IIOP protocol**. We can develop **Language independent** and **OS Independent Distributed applications** using CORBA.

Under communication between CORBA clients and CORBA services, method calls are passed to **Object Request Brokers** (ORBs). These ORBs communicate via the **Internet Inter-ORB Protocol (IIOP)**. IIOP transactions can take place over TCP streams, or via other protocols (such as HTTP), in the event that a client or server is behind a firewall. The following diagram shows a client and a servant communicating.

|  |  |
| --- | --- |
| corba.gif (7788 bytes) | corba2.gif (7735 bytes) |
| **CORBA client sends a request through its local ORB to a remote ORB's servant** | **CORBA servant sends back a response to a remote ORB** |

**Note:**

1. CORBA is from OMG (800+ companies except microsoft).
2. CORBA based technologies are :

* IDLJ from SUN MS (Interface Definition Language for Java).
* BOSS from IBM.
* Visibroker from Visigenic.

1. According to CORBA specifications

* Applications are platform, OS and language independent.
* The allowed languages are C, C++, java, Objective C etc.

**Note:**

**“CORBA is strong in specification wise and failed in implementation wise.”**

**Problems with CORBA in general:**

* The goals of being language-independent, platform-independent, and suitable for all distributed-systems development, along with a decade's worth of refinements and new features while maintaining backward compatibility, have led to a lot of complexity and bloat in the specifications and implementations.
* There is a steep learning curve. Rummaging through the mountains of specs to find the 5% that is relevant to your needs can be frustrating.
* Some of the language mappings seem "**unnatural**", leading to idiomatic CORBA-specific code that does not interface easily with standard libraries or third-party components. Many programmers have a hard time learning and understanding the rules of the mappings.
* Versioning and compatibility of interfaces between versions of your business server. (Mismatches cause memory corruption and/or crashing.)
* There is no inheritance for exceptions, so you cannot group common error types and handle them uniformly via a base exception class. This makes error handling a real pig, and not very extensible.
* Can be difficult to use if server and/or client is behind a firewall or if network address translation is being used.
* It is very difficult in setting up the distributed application behind the firewall.



**RMI**

RMI stands for **Remote Method Invocation**.

After CORBA, SUN released RMI as an API to support distributed programming. It allows a Java Object to be exposed over the network. SUN designed RMI keeping in view of all the complexities and drawbacks in CORBA and ensured those will not be reintroduced in RMI. So, in RMI java developer instead of writing a different language script like IDL, he will start with Java Object development and once he finish writing the business logic in the POJO, he will give it as input to RMI Compiler, which will **generates abstractions (a layer on top of it) to expose over the network**.

Here the programmer need not to worry to code a different language file or need not write the code in some other generated object, rather the development starts with POJO and once finishes will generate network abstractions to expose it.

To expose the generated object over network, it has to be deployed on a server called **RMI server**. The RMI server is **open source** and would be shipped as part of JDK and is very light weight server. With this if you observe almost all the dis-advantages with CORBA have been fixed in RMI and it is light weight.

**Out of its simplicity, it introduced certain dis-advantages as well.**

In RMI developer has to deploy the RMI object on RMI server and it is absolutely light weight, it is just a registry holding the RMI object with a name to locate it. Apart from this the server **would not provide any infrastructural services** like security or connection pooling or transactions and these has to be coded by developer itself, which would be so costly and huge amount of effort is required.

**If we want to implement RMI application, we need to provide minimum four components.**

a) Remote Interface

b) Remote Interface Implementation

c) Stub class for Client

d) Skeleton class for Server

Then register or publish our RMI application into RMI registry that means register our Remote Object with one central repository. So, that everyone can access it. Finally, write a sample client to test this application.

**Why RMI based Distributed applications are JAVA Specific? That means what makes it as Language Dependent?**

**If we observe any RMI based Distributed application:**

a) **Remote Interface and Remote Interface Implementation**

Remote interface contains methods to provide services. Both Remote interface and implementations are JAVA interface and classes only. They are specific to Java only.

b) **RMI registry**

A RMI registry is the software system that stores, organizes and provides access to information in a directory. We need to publish our RMI remote object into this RMI registry with some name. We need to use JNDI (Java Naming and Directory Interface) API to access this Directory Service to get the reference of that Remote object. But JNDI API is Java Specific that means only Java Clients can use this API.

c) **RMI Protocol**

RMI Applications uses JRMP protocol. It is also Java Specific Server uses Skeletons .Client uses Stubs to send a request to server side stub or to receive response from Server. These RMI Stubs and Skeletons are also Java specific.

The following three RMI components make RMI application as Language Dependent Distributed application.

** Remote Interface and Implementation**

** Directory Service**

** RMI Protocol**

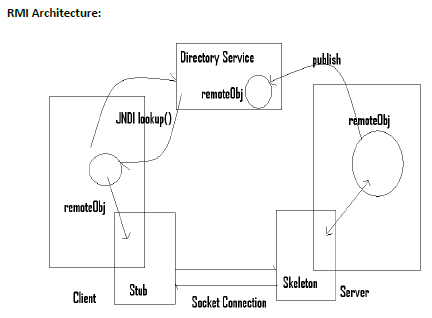
That means we can NOT implement inter-operable Distributed Applications using RMI technology. It is the major drawback of RMI API.

**What is Marshaling?**

That means whenever client wants to send a Request to a server, then it packs all arguments into an object and send it to server. This packing is called Marshaling.

**What is Unmarshaling?**

Once server receives Marshaled object, it unpacks that object and calls respective service call. This unpacking is called Unmarshaling.



**Advantages of RMI:**

 Platform Independent because it is implemented in JAVA

**Drawbacks of RMI:**

 Language Dependent

Not firewall free

**Note:**  
  
**POJO** **:**

Plain Old Java Object is a Java object not bound by any restriction other than those forced by the Java Language Specification.  
An object that compiles under JDK can be considered a Plain Old Java Object. No app server, no base classes, no interfaces required to use.  
POJO is a Java object not bound by any restriction other than those forced by the Java Language Specification.

**Properties of POJO**

* All properties must public setter and getter methods
* All instance variables should be private
* Should not Extend prespecified classes.
* Should not Implement prespecified interfaces.
* Should not contain prespecified annotations.
* It may not have no argument constructor

**Example of POJO**

*public class POJO {*

*private String value;*

*public String getValue() {*

*return value;*

*}*

*public void setValue(String value) {*

*this.value = value;*

*}*

*}*

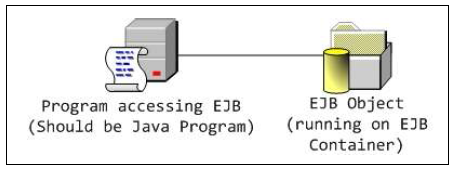
**EJB**

EJB stands for **Enterprise Java Bean**, which is next to RMI.

From the EJB’s onwards the concept of **managed object** came into picture. You will write an Object which acts as an EJB and is deployed on a managed server called **EJB container**. While your EJB object is executing, it might need some external infrastructural resource support, instead of programmer coding for it, like security or transaction or auditing or connection pooling, these would be configured, maintained and provided to the EJB object by EJB Container itself.

So, programmer need not worry about implementing these infrastructural resources and he cannot concentrate on his business logic leaving it to Container to provide them. This makes programmer life easier rather than coding each and everything out of box.

But the problem with EJB is when you expose an Object as EJB, it can be accessed over the network by other programs. **The other programs which want to access the EJB object should also be Java applications only and non-java programs cannot access EJB’s as shown below**.



The concept of managed objects became popular with the introduction of EJB.

**EJB are of three types:**

**Session beans,**

**Entities and**

**Message driven beans.**

When you build an EJB and deploy it on an Application Server it will become a managed bean. So, all the external dependencies like connection pooling, transactions etc. are provided by underlying infrastructure and programmer is freed up from coding these crossing cutting features.

**Advantages of EJB**

 Managed Objects

 Container will provide all the external dependencies

 Equipped with Enterprise class features

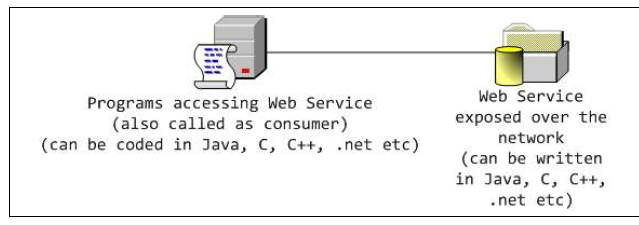
**Dis-advantages of EJB**

 Specific to Java, non-java clients can’t access EJB object

**Web services**

Web services also allows a program to expose objects over the network, but the difference between other distributed objects (RMI or EJB) and Web Service distributed object is these are not only platform (OS) independent, these are even language independent. Which means you can write a Web service Object using C, C++, Perl, Python, PHP, Java or .Net and can expose over the network. In order to access this, the other program could be written in any of the programming languages like C or C++or Perl or Java etc.

Anything that is accessible irrespective of Platform and programming language is called **Interoperability**. This mean we are building distributed interoperable programs using Web services.



**What is Interoperability?**

Interoperability is the ability of diverse systems to work together (inter-operate).

**Advantages of Interoperability**

* Connect existing software.
* Easy to extend the existing system

**Web Services Use Cases:**

**Example 1:**

If we want to book a rail ticket, we can go for IRCTC website. Apart from IRCTC website, MakeMyTrip and Paytm are also providing facility to reserve a ticket by partnering with IRCTC. Here IRCTC is called **Business Owner** and MakeMyTrip and Paytm are called **Business Partners**.

Here, how MakeMyTrip/Paytm is getting the tickets availability or ticket status of the train ticket? Will IRCTC will provide the complete database connection details like user name, password, connection details etc…? Is it possible? The answer is **NO**. Then security issue occurs, if the DB details are being exposed to business partners.

If there are two customers from IRCTC website and MakeMyTrip/Paytm website are trying to book same ticket, then it should not allow. So, we should have a common source of information, it should be reflected to all Business owners and Business partners.

Not only the above problem, in some case if IRCTC wants to change the railway reservation procedure. For example he wants to change the **Tatkal** ticket booking procedure; it cannot again go to its Business partners to change the code accordingly.   
  
In all these kinds of scenarios, the IRCTC will provide its facility to all the business partners by using **WEB SERVICES**.

**Example 2:**

During 10th Class results, the SSC board will allow few other websites (like Manabadi.com) also to display the results to the students. In this case SSC is the Business Owner and manabadi is the Business Partner. SSC will help manabadi by using Web services only.

**Example 3:**

**Smart Refrigerator:** In this case we have to program the fridge in such a way like Tomato must be minimum of 1 kg in the fridge. During the normal daily usage, if the tomato availability in the fridge goes below 1 kg, then internally the message/call will be passed to the nearby store and store will take the request and sends the tomato to our home. Internally it works using Web Services only. Internally it uses IOT too.

Not only these, many other things like Cloud Genie, Amazon Buttons and many things works using web services technologies only.

**A Web Service is can be defined by following way:**

Web service is a way of communication that allows interoperability between different applications on different platforms is a software system for interoperable machine to machine communication.

It is a collection of standards or protocols for exchanging information between two devices or application.

**Let's understand it by the figure given below:**



As you can see in the figure, java, .net or PHP applications can communicate with other applications through web service over the network. For example, java application can interact with Java, .Net and PHP applications. So web service is a language independent way of communication.

The communication can be done through a set of XML messages over HTTP protocol.  
  
Web services are browsers and operating system independent service, which means, it can run on any browser without the need of making any changes. Web Services take Web-applications to the Next Level.  
  
The World Wide Web Consortium (W3C) has defined the web services. According to W3C, “**Web Services are the message-based design frequently found on the Web and in enterprise software. The Web of Services is based on technologies such as HTTP, XML, SOAP, WSDL, SPARQL, and others.”**

Let’s say, you are a java developer and you can publish your functions on internet or LAN through java web service. So, any other developer (let’s say .Net developer) can access your function.

**Micro Soft COM/DCOM**

If we use Micro Soft DCOM to implement our distributed applications, then that become OS dependent application. That means client must use Windows OS only. To overcome all these problems, some organizations like IBM, Micro Soft, Sun Micro Systems etc. have worked on it and introduced a standard that is Web service. Using Web services, we can implement the server and client in any language or in any platform (OS). As the web services uses HTTP protocol, it is very easy to developer the applications running behind the fire wall.

**Layman Terms about architecture/Components of web services:**

Here, person P1 and P2 wants to communicate with each other. So, we need the following things:  
  
**a. Medium: [Network- Technically]**  
A perfect medium to communicate may be mobile.

b. **Some Rules: [Protocol - Technically]**  
For effective communication, we should have some rules like till the time person P1 stops talking, P2 should not respond and vice versa. Then only communication will be effective.  
  
c. **Common Language:** **[Common Character set encoding]**  
If P1 knows Chinese and P2 knows Telugu only, they can’t communicate properly. They should have come common.

**Architecture**

Let us try to understand what we need to build a Web Service in general. It’s similar to block diagram of a computer like we need mouse, key board, monitor, CPU etc. Let us also identify here what are all the various components required to build a Web Service.

**1. Medium**

If two people want to exchange information between them, what they need? To propagate their wave lengths from one person to other, they need medium.

Medium acts as a physical channel on which data would be exchanged. If two computers want to exchange data, first they should be connected over a wire or network.

Medium -   
**Network**

**Provider**

**Consumer**

**Note:**

**Client => Consumer**: a client is the one who consumes.

**Server => Provider**: the one who produces something to the client.

**Client / Server** is often used in architectural concepts, so it does not describe the WS itself, but the architectural pattern between the one on which you connect and the one who connect. **Consumer/Provider** is more understandable since it describes the service architectural pattern. You have a part who gives a service, and another part who uses it.

[](http://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwibi_iSt77QAhVH7CYKHah9B1AQjRwIBw&url=http://www.javatpoint.com/service-oriented-architecture&psig=AFQjCNFH_ROxwgdWRF_oWFv82pII73cQ7Q&ust=1479975079213161)

**Note**: The Provider should be developed first then consumer.

**2. Protocol**

Having medium itself is enough? With medium we can propagate the wavelengths, but there is not guarantee of receiving the information intact and properly. Let’s say two people are talking to each other, instead of one person listening to other both the people started talking to each other at the same time.

It seems instead of talking there is barking at each other, would the communication be effective? Rather if there are set of rules that guards their communication, for e.g. first person says am starting the conversation so that the other person would be in listening mode, once the first person finishes, he will say am done so, that the other person can start the conversation by following the same rules.

This indicates we need set of rules that guards the communication channel which is nothing but a protocol. If two computer systems want to exchange data, having network itself is not enough we need a protocol to guard the exchange of data over them. Protocol simulates that everyone has their independent bands of communication channel to exchange information, even though everyone is sharing the same physical channel. In the same way if two programs want to exchange the data over the network, they need physical data transport protocol like TCP/IP or RMI/IIOP or HTTP or SMTP etc. The recommended protocol in case of Web Service communication is HTTP.

Protocol -   
**HTTP**

**Consumer**

**Provider**

Medium -   
**Network**

**Layman Terminology:  
Raw representation of data:**Between the consumer and server, we have to we must not transfer the data in terms of “**Raw representation of data**”. Because the computers can only accept the data in bytes.

M 1   
[Raw Representation of data:  
Hi! How Are You ?]

M2- [It cannot understand]

Here, internally if we transfer the data in the form of characters also, the machine internally converts those “Raw Representation of Data” to sequence of characters. Then again it converts those characters to the corresponding numeric values.

**For Example [Let’s assume] :**

a🡪25, A🡪65, B🡪66, \*🡪134,? 🡪 239

All these numeric numbers must be again converted to Byte format.

25🡪 11001  
65🡪 1000001  
66🡪1000010  
134🡪 10000110

239🡪 11101111

Here, if you see the above binary values, you can see that some binary values have 5 characters, some have 6, some 7 and remaining have 8 binary values. So, as per receiver end, the M2 machine cannot understands whether how much binary date it should accept the value from the M1 and construct the character for that. In this case, we should have some criteria. We should make a uniform length for each character i.e 8. So, we have to append 0’s in the starting.

25🡪 00011001  
65🡪 01000001  
66🡪 01000010  
134🡪 10000110

239🡪 11101111

So, then the receiver will understand the values and considers 8 places and constructs it as 1 character.

So, companies defined few standards like ASCII, UTF-8, and UTF-16 etc. So, the M1 and M2 both should understand the common language for transmission.

CSV [**comma separated values**]:

M 1   
[CSV:  
1, 497, Srinivas,B, male]

M2- [It cannot understand]

Here M1 is trying to pass the data to the M2 by using CSV format. But, it is not possible. As there is no structure and semantics surrounded to the data. Here M2 might be confused to know whether 1 is serial number or roll number. There are no proper semantics to this. There is no boundary also. If user enters “Srinivas,B” as one field, then as CSV is a comma separated description, it cannot predict the boundaries also. Whether Srinivas is the values or Srinivas,B is the value, it cannot predict. So, no proper boundaries are established here.

**XML:**

In XML we are sending data in a plain test and it is very easy to understand. Here we are sending data in structured and proper semantics are being attached to the data also. So, xml is perfect to transfer the data.

M2  
<info>  
 <slno>1<slno>  
 <rollno>497<rollno>  
 <name>Srinivas,B<name>  
 <gender>male</gender>  
</info>

M1  
<info>  
 <slno>1<slno>  
 <rollno>497<rollno>  
 <name>Srinivas,B<name>  
 <gender>male</gender>  
</info>

**Why not Excel or Pdf format to transfer the data?**

Excel works only with Microsoft related things. It doesn’t support interoperability.  
pdf is not a programmable language.

**3. Language**

Now we have medium and protocol to transmit information between two people, but both the parties has to exchange the information in same language, which means a common language to exchange information in an understandable manner like English or French or Japanese etc.

If these two are computers, we can exchange the data between them in an ASCII or UNICODE encoded characters. For example it could be a CSV or a text file of information encoded in common encoding.

CSV files are good when your data is strictly tabular and you know its structure. If you have relationships between different levels of data, CSV are not competent to hold such information. CSV is not self-explanatory which means based on the position of fields user has to interpret the type or nature of information. Let’s consider a simple example to understand the same.

There are two systems, one is a Bank system and the other is client who is trying to send the deposit data to deposit into his account. He has sent the information in a **CSV\*** format as shown below.

1, AC342, Suresh, 422, D, W

\*A **CSV** is a **comma separated values** file, which allows data to be saved in a table structured format.

The field order in which he sent the information is Serial No, Account No, Name, Amount, Operation Type and Branch Code. The Bank system upon receiving the data has started interpreting the data in this order Amount, Account No, Name, Serial No, Branch Code and Operation Type.

The problem with the above representation is the data is not self-explanatory, which is nothing but semantics of data is missing; the receiver of the data also has to interpret the data in same order (which is not guarantee). This might lead to incorrect operations as above instead of deposit, it leads to withdrawal.

Instead of CSV, we can represent the data in **MS-Word, MS-Excel** etc. But the problem with these kinds of representations is those are recognized by only Windows and few other platforms, which mean those, are platform dependent formats. We are trying to build an interoperable solution, which means the data we carry between systems must also be interoperable.

**Only way of carrying data between systems in an interoperable manner is XML**.

XML Stands for Extensible Markup Language. It is not a programming language; it is used for storing information in it. How we store the data in MS-Word or MS Excel or a CSV file similarly XML is also a document which stores data in a **XML Format**. The data represented in XML has well-defined structure and has semantics attached to it (self-explanatory) as shown below.

*<info>*

*<sno>****1****</sno>*

*<accno>****AC324****</accno>*

*<name>****Suresh****</name>*

*<amount>****422****</amount>*

*<operationType>****D****</operationType>*

*<branchCode>****W****</branchCode>*

*</info>*

In the above XML it is clear that which data is conveying what type of information, so that the receiver of this will not have a chance of miss-interpreting the information. So, the defacto (in fact; in reality) standard for exchanging information between computers is XML.

**Consumer**

**HTTP**

**Provider**

**Network**

**4 Binding Protocol**

Having XML is not enough to exchange data between two systems. We need a layer on top of it to classify the information.

For e.g. we not only send the **business data** for communication, along with that we might need to send transport specific or some other **helper information**. Using XML we cannot differentiate such kind of information.

Consider the below example where the user wants to register his account online, so he is sending the registration information in a XML format to the banking system. Registration data will generally contains AccountNo, UserName, Password, Re-Password, Secret Password provided by bank to register.

*<registration>*

*<accountNo>****AC222****</accountNo>*

*<password>****533fd****</password>* ***\_ Secret password***

*<username>****John****</username>*

*<password>****33kk****</password>* ***\_ Password with which want to register***

*</registration>*

In the above XML we are trying to represent two different types of data one is business data nothing but AccountNo, username and password with which you want to register your account. The other one is helper data or processing data secret password given by bank with which you can be authorized to register.

As discussed it contains two types of data, it would be confusing for the person who is trying to read this XML, to identify which is processing data and which is business data. We need a wrapper on top of it to differentiate the same, which is nothing but SOAP.

SOAP stands for “**Simple Object Access Protocol**”. It is also an XML, but SOAP XML has a **rigid structure and has reserved grammar** with which we need to write it.

SOAP XML has **root element as Envelope** and has two sections, **first one is header and second one is body** as shown below.

*<Envelope>*

*<Header>*

***<!—processing data or helper data -->***

*</Header>*

*<Body>*

***<!—Business data -->***

*</Body>*

*</Envelope>*

All the processing data will be placed under <Header> element and the business data will be placed inside <Body> element.

SOAP is also called as **Binding Protocol**, because it binds our business XML.

It acts as an **Application Specific Protocol**, because the receiver of the SOAP XML will be able to identify and differentiate which is processing data and which is business XML. SOAP defines the rules in which other applications have to interpret the data so it is called **Application Specific protocol.**

Along with that SOAP is also called as **carrier of the actual Payload**. As our business XML is placed inside the SOAP XML and is carried by SOAP. It is also called as carrier of the actual payload.

**Consumer**

**Provider**

**Network**

**HTTP**

**5. WSDL (Web Service Description Language)**

As we know, in a Web Service two parts are involved, one is provider and the second one is consumer. Provider is the person who always listens for incoming requests from a consumer and serves data as part of request. Consumer is the requestor, who always seeks information from the Provider.

So, for a consumer in-order to get the data from the Provider, he needs to know the information about the provider. This means the information about the provider has to be document in some document like MS-Word or PDF etc.

The problem with this is you are trying to build an Interoperable distributed application which can be accessible by any one literally, but the documentation about it is being documented in non-interoperable document, will it be understood by everyone. That means not only the program the documentation should also be interoperable, there comes WSDL.

WSDL Stands for **Web Service Description Language**. It is also an XML type document, this makes it interoperable. WSDL has a pre-defined structure and has grammar, using which you need to write the WSDL document. WSDL document will provides the entire information about the provider like Service name, Number of Operations, their parameters and return types and Location of the Service etc.

The consumer can know the entire information about the provider by seeing its WSDL document.

**Note**: WSDL is a static entity which describes information about the Provider. It is not a program to execute.

Documentation- **WSDL**

**Consumer**

**Provider**

**Network**

**HTTP**

**6. UDDI (Universal Description and Discovery Integration**

**Registry)**

After documenting the information about the service in WSDL, we need to distribute the WSDL document to the consumer’s. Do we have one consumer for a Provider and multiple Consumers?

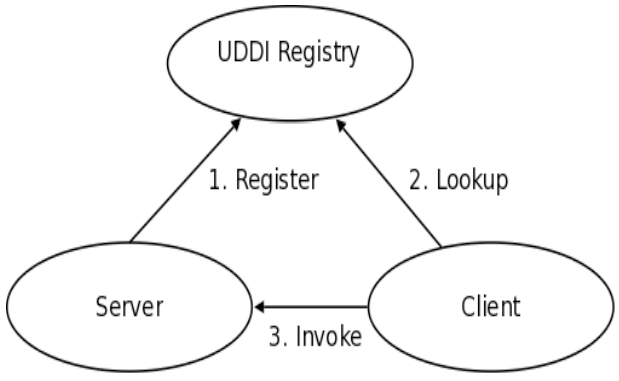
For a Provider it could have as many consumers as possible, so we need to circulate or distribute the WSDL document to each and every consumer. Will it be feasible to give WSDL document to each and every consumer, ideally speaking will not be possible that’s where UDDI came into picture.

UDDI stands for “**Universal Description and Discovery Integration Registry**”, this is the registry which store’s all the WSDL documents in it. Provider will **publishes** the WSDL documents to the UDDI registry and Consumers will browse for a WSDL document from the registry.

UDDI should be accessible irrespective of programming language which means these should also be interoperable, that’s why those are built using XML technology, so those are also called as XML registries.

UDDI is an [XML](http://searchsoa.techtarget.com/definition/XML)-based registry for businesses worldwide to list themselves on the Internet. Its ultimate goal is to streamline online transactions by enabling companies to find one another on the Web and make their systems interoperable for e-commerce. UDDI is often compared to a telephone book's white, yellow, and green pages. The project allows businesses to list themselves by name, product, location, or the Web services they offer.

**General Invocation Pattern using the UDDI Registry**



* **In step 1**, it is shown how a business publishes services to the UDDI registry.
* **In step 2**, a client looks up the service in the registry and receives service binding information.
* Finally **in step 3**, the client then uses the binding information to invoke the service.

**UDDI is an XML-based standard for describing, publishing, and finding web services.**

* UDDI is a specification for a distributed registry of web services.
* UDDI is a platform-independent, open framework.
* UDDI can communicate via SOAP, CORBA, and Java RMI Protocol.
* UDDI uses Web Service Definition Language (WSDL) to describe interfaces to web services.
* UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services.
* UDDI is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet.

**UDDI has two sections:**

* A registry of all web service's metadata, including a pointer to the WSDL description of a service.
* A set of WSDL port type definitions for manipulating and searching that registry.

**There are two types of UDDI registries**

* **Private UDDI** – These are maintained internal to an organization and would be accessible within the organizational n/w.
* **Public UDDI** – These are accessible globally across over the network. But few would be open source hosting and few would be commercial.

**Private UDDI Registries**

As an alternative to using the public federated network of UDDI registries available on the Internet, companies or industry groups may choose to implement their own private UDDI registries.

These exclusive services are designed for the sole purpose of allowing members of the company or of the industry group to share and advertise services amongst themselves.

Regardless of whether the UDDI registry is a part of the global federated network or a privately owned and operated registry, the one thing that ties them all together is a common web services API for publishing and locating businesses and services advertised within the UDDI registry.

**Summary:**

From the above we understood what are the components required to build a Web

Service, below is the diagram depicting the Architecture of a Web Service.

**Publishing**

**Consumer**

**Provider**

**Network**

**HTTP**

Documentation- **WSDL**

**UDDI Registry**

**Discovery**

**Overall Explanation**:  
  
Consumer is the person who tries to access the information from the Provider, and Provider is the person who always services the Consumer.

Those should be often referred to as Consumer and Provider and should not call them as Client and Server in case of Web Services.

Consumer and Provider want to exchange information, so first they should be connected with a wire and needs a protocol to guard their communication. The recommended protocol in a web service communication is HTTP. **HTTP protocol is firewall friendly**.

They need a common language to exchange data; the defacto standard for exchanging information between computers is XML. But only XML is not enough, as a Web Service communication not only transmits business data, it might also transmit helper or processing information, so we need a layer on top of it called SOAP.

SOAP stands for “**Simple Object Access Protocol**” acts as a binding protocol to classify the information.

If the consumer wants to access the provider, he needs to know the information about the provider. So, the information about the provider is documented in a document called WSDL and is being published by the Provider in a registry called UDDI. The process of putting the document into the UDDI registry is called

Publish and would be generally done by Provider.

Now the consumer has to connect to the registry and performs a search on the registry which is called Discovery to find an appropriate WSDL document.

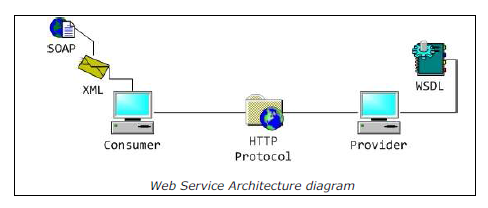
Downloads it locally and now starts understanding the information about provider.

He will builds a program called “Web Service Client” using the information in the

WSDL document through which he will sends the request and gets the response back.

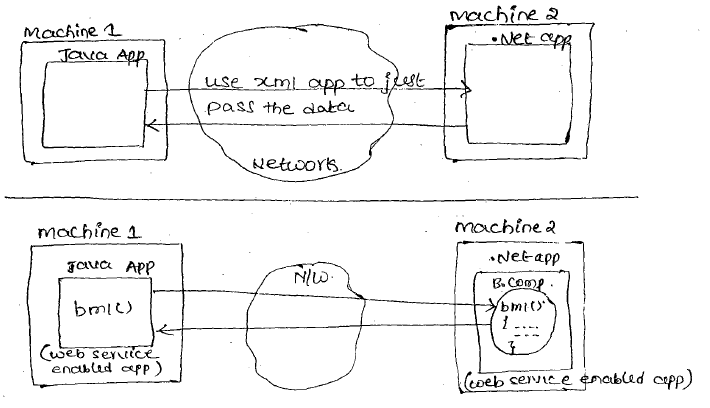
By the above we understood that language for exchanging the information is XML.

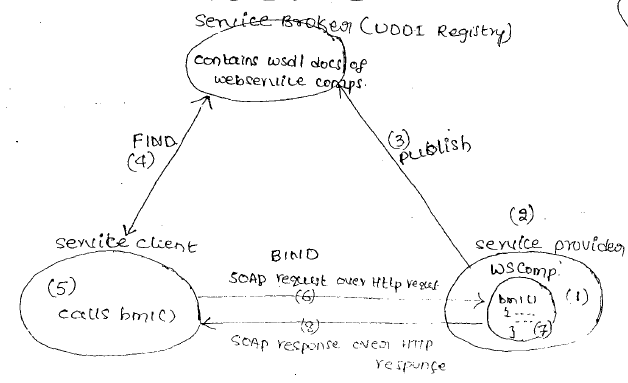
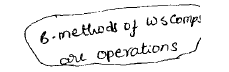
Binding protocol is SOAP which is nothing but XML and Description of the Service is WSDL, is also XML. Registry in which we store the WSDL is UDDI, built on top of XML technologies, everything in Web service is built on XML and without XML Web Services would not be possible, so first we need to understand XML, let’s move to XML.



**Note:**

1. If we want to pass the data in terms of incompatible applications, then we must pass the data in terms of xml files.
2. If we want to see interaction between two incompatible applications then they must be developed in web services enabled applications.



1. **The overview diagram of webservices:** 
   1. Web service is a XML, SOAP; HTTP based distributed technology that allows us to develop inter-operable distributed applications by using technology based of programmer choice.
   2. All hardware components are inter-operable (works everywhere) without working about compatibility.
   3. To develop inter-operable software components use web services that means we can us .Net components in Java and vice versa.

**W.R.T. Diagram**

**1**. Service provider develop web service component as web application, it is the choice technology (.Net, java) having Business methods.

**2.** Service provider generates WSDL does for web service component having the details about components and business methods (xml entries).

**3.** Service provider publishes the WSDL document in UDDI registry for global visibility.

**4.** Service client gathers business component from UDDI registry.

**5.** Service client understands WSDL document and develops client application in his choice technology. This client application calls Business methods of web service components.

**6. 7.8.** Web service client application calls the Business method of the web service components and gathers the result by using the SOAP over HTTP Protocol.

**Web Services components**

 **SOAP** (Simple Object Access Protocol)

 **UDDI** (Universal Description, Discovery and Integration)

 **WSDL** (Web Services Description Language)

 **HTTP** (Hyper Text Transfer Protocol)

All these Web Services four standards are from W3C organization. W3C Stands for the World Wide Web Consortium.

1. **SOAP:**

SOAP is an XML-based protocol for exchanging information between computers.

* SOAP is a communication protocol.
* SOAP is for communication between applications.
* SOAP is a format for sending messages.
* SOAP is designed to communicate via Internet.
* SOAP is platform independent.
* SOAP is language independent.
* SOAP is simple and extensible.
* SOAP allows you to get around firewalls.
* SOAP will be developed as a W3C standard

1. **UDDI**

UDDI is an XML-based standard for describing, publishing, and finding web services.

* UDDI stands for Universal Description, Discovery, and Integration.
* UDDI is a specification for a distributed registry of web services.
* UDDI is platform independent, open framework.
* UDDI can communicate via SOAP, CORBA, and Java RMI Protocol.
* UDDI uses WSDL to describe interfaces to web services.
* UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services.
* UDDI is an open industry initiative enabling businesses to discover each other and define how they interact over the Internet.

**WSDL**

WSDL is an XML-based language for describing web services and how to access them.

* WSDL stands for Web Services Description Language.
* WSDL was developed jointly by Microsoft and IBM.
* WSDL is an XML based protocol for information exchange in decentralized and distributed environments.
* WSDL is the standard format for describing a web service.
* WSDL definition describes how to access a web service and what operations it will perform.
* WSDL is a language for describing how to interface with XML-based services.
* WSDL is an integral part of UDDI, an XML-based worldwide business registry.
* WSDL is the language that UDDI uses.
* WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'.

**HTTP:**

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This is the foundation for data communication for the World Wide Web (i.e. internet) since 1990. HTTP is a generic and stateless protocol which can be used for other purposes as well using extensions of its request methods, error codes, and headers.

Basically, HTTP is a TCP/IP based communication protocol, that is used to deliver data (HTML files, image files, query results, etc.) on the World Wide Web. The default port is TCP 80, but other ports can be used as well. It provides a standardized way for computers to communicate with each other. HTTP specification specifies how clients' request data will be constructed and sent to the server, and how the servers respond to these requests.

**Basic Features**

**There are three basic features that make HTTP a simple but powerful protocol:**

* **HTTP is connectionless:**

The HTTP client, i.e., a browser initiates an HTTP request and after a request is made, the client disconnects from the server and waits for a response. The server processes the request and re-establishes the connection with the client to send a response back.

* **HTTP is media independent:**

It means, any type of data can be sent by HTTP as long as both the client and the server know how to handle the data content. It is required for the client as well as the server to specify the content type using appropriate MIME-type.

* **HTTP is stateless:**

As mentioned above, HTTP is connectionless and it is a direct result of HTTP being a stateless protocol. The server and client are aware of each other only during a current request. Afterwards, both of them forget about each other. Due to this nature of the protocol, neither the client nor the browser can retain information between different requests across the web pages.

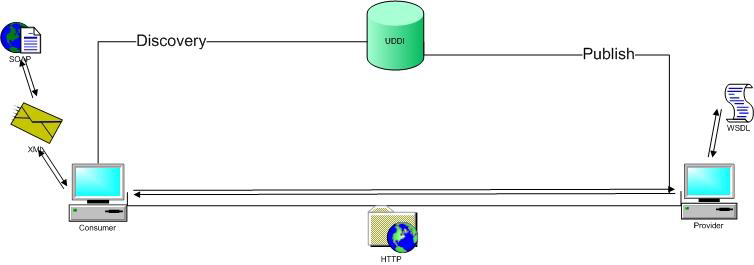
HTTP/1.0 uses a new connection for each request/response exchange, where as HTTP/1.1 connection may be used for one or more request/response exchanges.

**A typical Web Service invocation**

***Web Service: Weather Service***

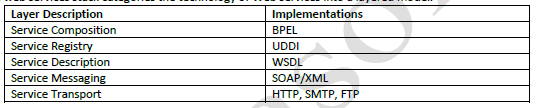
****

**The Web Service Architecture**



**Web Service Stack**

The web services stack categories the technology of Web services into a layered model.



**Definitions:**

**Simple Object Access Protocol(SOAP) :**

SOAP is a protocol specification for exchanging structured information in the implementation of Web services in computer networks. It relies on XML as its message format.

**Web Service Description Language(WSDL) :**

WSDL stands for Web Service Description Language. It is an XML file that describes the technical details of how to implement a web service, more specifically the URI, port, method names, arguments, and data types. Since WSDL is XML, it is both human-readable and machine-consumable, which aids in the ability to call and bind to services dynamically.

**Universal Description, Discovery and Integration (UDDI)** :   
  
UDDI stands for Universal Description, Discovery and Integration. It is a directory service. Web services can register with a UDDI and make themselves available through it for discovery.

**Multiple definitions of Web Services:**

Different books and different organizations provide different definitions to Web Services. Some of them are listed here.

* A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. **For example**, a client invokes a web service by sending an XML message, and then waits for a corresponding XML response. As all communication is in XML, web services are not tied to any one operating system or programming language--Java can talk with Perl; Windows applications can talk with UNIX applications.
* Web services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.
* Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.
* A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

**To summarize, a complete web service is, therefore, any service that:**

* Is available over the Internet or private (intranet) networks
* Uses a standardized XML messaging system
* Is not tied to any one operating system or programming language
* Is self-describing via a common XML grammar
* Is discoverable via a simple find mechanism

**To develop all the resources of the web service enabled application in Java**

1. Use JAXR API to make service provider and service client interacting the UDDI registry.
2. Use JAX-RPC/JAX-WS to develop web service components and clients.
3. Use JAX-B to get the interaction between web service client and component.

**Note:**

As of now JAVA, J2EE and .Net are the technologies that work with web services.

**How Does a Web Service Work?**

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP.

A web service takes the help of:

* **XML to tag the data**
* **SOAP to transfer a message**
* **WSDL to describe the availability of service.**

You can build a Java-based web service on Solaris that is accessible from your Visual Basic program that runs on Windows.

You can also use C# to build new web services on Windows that can be invoked from your web application that is based on Java Server Pages (JSP) and runs on Linux.

**Example**

Consider a simple account-management and order processing system. The accounting personnel use a client application built with Visual Basic or JSP to create new accounts and enter new customer orders.

The processing logic for this system is written in Java and resides on a Solaris machine, which also interacts with a database to store information.

**The steps to perform this operation are as follows:**

* The client program bundles the account registration information into a SOAP message.
* This SOAP message is sent to the web service as the body of an HTTP POST request.
* The web service unpacks the SOAP request and converts it into a command that the application can understand.
* The application processes the information as required and responds with a new unique account number for that customer.
* Next, the web service packages the response into another SOAP message, which it sends back to the client program in response to its HTTP request.
* The client program unpacks the SOAP message to obtain the results of the account registration process.

**Web Applications Vs Web Services:**

In simple words, services that can be accessed over network are called **web services**. So how does it differ from web application, they are also services that are accessed over network.

**There are few attributes that clarifies this difference.**

* Web applications are meant for users and to be accessed in browser having human readable format whereas web services are meant for applications to access data in the format of XML, JSON etc.
* Web applications always use HTTP/HTTPS protocol whereas traditional web services use SOAP protocol. Recently REST is getting popularity that is an architecture style and almost all times run on HTTP/HTTPS protocol.
* Web applications are not meant for reusability whereas this is one of the benefits of web services. A single web service can be used by different kinds of applications.
* Web application can access web services to access some data or to perform some tasks; web services can’t access web applications to fetch some data.
* Web applications are capable to maintain user session, web services are stateless.

**Evolution**

Initially every language can build programs which can run on one machine. There after every program language strived to build applications which can not only talk to the applications running on same machine but also they should be able to communicate with programs running on different machines but, connected over the network, which is nothing but distributed programming. With this the invention didn’t end, they want to still find a way to build distributed programs which can talk over heterogeneous systems (irrespective of platforms and programming languages) is called **interoperable distributed applications**.

To build interoperable distributed applications, every software vendor in the industry started defining their own standards. Let us say SUN has defined its own standards, Microsoft defined its own standards and IBM came with its own standards etc.

|  |  |  |  |
| --- | --- | --- | --- |
| **Vendor** | **SUN** | **Microsoft** | **IBM** |
| Transport Protocol | HTTP | FTP | SMTP |
| Language for exchanging data | XML | MSXML | CSV |
| Description | JWSDL | MSWSDL | IBM4SDL |

**If vendor has their own stack of standards for example as shown below:**

Java programs can only understand the data transmitted over HTTP, so if a

Microsoft program is sending the data over FTP; it cannot be read/received by a Java application.

In this case Interoperability never will become a reality. Finally it was realized by every software vendor that we cannot build independently an interoperable distributed applications, unless everyone has agreed upon same standards.

There comes an organization **WS-I, stands for Web Service Interoperability** **organization**. It is a non-profitable organization which is formed to build open standards for building interoperable distributed applications. The members of this group are people/representatives from various vendors in the software industry.

**WS-I has released BP 1.0 (Basic Profile) specification document**. The specification document comprises of **set of guidelines or standards** that must be used to build an interoperable distributed programs. BP 1.0 doesn’t talk about how to develop an interoperable distributed application in Java rather they will talk about the standards or guidelines to build web service in any programing language. After few years of BP 1.0, WS-I has again released one more specification document BP 1.1 which is the successor of BP 1.0 which contains some advancements than its earlier.

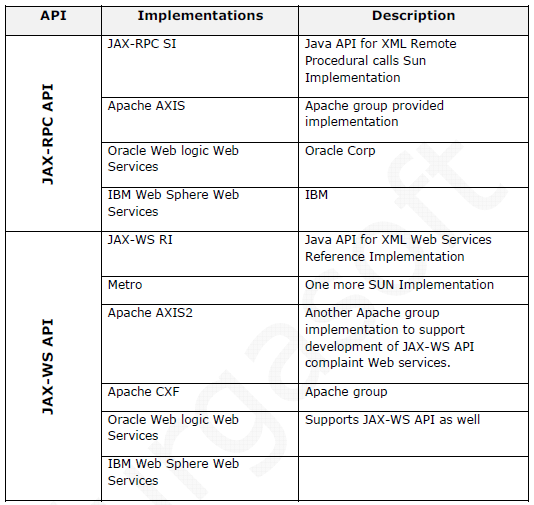
**Java API’s for WS-I Specification**

As discussed above WS-I’s BP 1.0 and BP 1.1 specification documents contains guidelines for building Web Services in any programming language, those are not specific for Java. SUN has to adopt those guidelines to facilitate building of Web services in Java.

To build WS-I’s BP 1.0 complaint Web Services SUN has released JAX-RPC API and to build BP 1.1 complaint Web Services JAX-WS API has been released. As BP1.1 is the successor of BP 1.0, similarly JAX-WS API is the successor of JAX-RPC API.

Always using Specifications and API’s we cannot build programs rather for an API we need implementation. There a lot of Implementations available for both JAXRPC API and JAX-WS API, few are open source and few other are commercial.

Below is the table showing the implementations of JAX-RPC API and JAX-WS API.



**In any technology based distributed application development the following resources are required:**

**a. Service Provider Application:**

It is server application (one or more) having the business logic based Business components / Business objects.

In web service environment it can be java or .Net.

1. **Service Client:**

The client application that calls the business methods of business components belonging to server application.

In web service environment this can be developed in Java or .Net.

1. **Service Interface:**

The common understanding document between service provider and service client application containing the declaration of business methods.

In web service environment it is WSDL (XML).

1. **Registry:**

The mediator between the service client and service provider having Business component details.

In web service environment the registry is UDDI [Universal Discovery, Description and integration]. (XMLRegistry).

**Advantages of Web Services**

**Exposing the Existing Function on the network**

A web service is a unit of managed code that can be remotely invoked using HTTP, that is, it can be activated using HTTP requests. Web services allows you to expose the functionality of your existing code over the network. Once it is exposed on the network, other application can use the functionality of your program.

**Interoperability**

Web services allow various applications to talk to each other and share data and services among themselves. Other applications can also use the web services. For example, a VB or .NET application can talk to Java web services and vice versa. Web services are used to make the application platform and technology independent.

**Standardized Protocol**

Web services use standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description, and Service Discovery layers) use well-defined protocols in the web services protocol stack. This standardization of protocol stack gives the business many advantages such as a wide range of choices, reduction in the cost due to competition, and increase in the quality.

**Low Cost of Communication**

Web services use SOAP over HTTP protocol, so you can use your existing low-cost internet for implementing web services. This solution is much less costly compared to proprietary solutions like EDI/B2B. Besides SOAP over HTTP, web services can also be implemented on other reliable transport mechanisms like FTP.

**Apart from the above, we have…**

 Platform Independent

 Language Independent

 Hardware and Software Independent

 Firewall Free

 Develop Reusable application-components

 Easy of Deployable

Loosely coupled

 Web enabled

**Disadvantages of Web Services**

Although the simplicity of Web services is an advantage in some respects, it can also be a hindrance. Web services use plain text protocols that use a fairly verbose method to identify data. This means that Web service requests are larger than requests encoded with a binary protocol. The extra size is really only an issue over low-speed connections, or over extremely busy connections.

 Although HTTP and HTTPS (the core Web protocols) are simple, they weren't really meant for long-term sessions. Typically, a browser makes an HTTP connection, requests a Web page and maybe some images, and then disconnects. In a typical CORBA or RMI environment, a client connects to the server and might stay connected for an extended period of time. The server may periodically send data back to the client. This kind of interaction is difficult with Web services, and you need to do a little extra work to make up for what HTTP doesn't do for you.

The problem with HTTP and HTTPS when it comes to Web services is that these protocols are "stateless"—the interaction between the server and client is typically brief and when there is no data being exchanged, the server and client have no knowledge of each other. More specifically, if a client makes a request to the server, receives some information, and then immediately crashes due to a power outage, the server never knows that the client is no longer active. The server needs a way to keep track of what a client is doing and also to determine when a client is no longer active.

Typically, a server sends some kind of session identification to the client when the client first accesses the server. The client then uses this identification when it makes further requests to the server. This enables the server to recall any information it has about the client. A server must usually rely on a timeout mechanism to determine that a client is no longer active. If a server doesn't receive a request from a client after a predetermined amount of time, it assumes that the client is inactive and removes any client information it was keeping. This extra overhead means more work for Web service developers.

**Note points on Web services:**

* From open community.
* It is web based distributed technology. (The server application of distributed technology must be a web application).
* Language Independent.
* Platform independent.
* Architecture independent.
* Allows converting any technology based application to web service application.
* Uses HTTP and SOAP Protocol.
* Travels the data in the form of XML content.

**Web services have the following special behavioral characteristics:**

**XML-Based**

Web Services uses XML at data representation and data transportation layers. Using XML eliminates any networking, operating system, or platform binding. Web Services based applications are highly interoperable application at their core level.

**Loosely Coupled**

A consumer of a web service is not tied to that web service directly. The web service interface can change over time without compromising the client's ability to interact with the service. A tightly coupled system implies that the client and server logic are closely tied to one another, implying that if one interface changes, the other must be updated. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.

**Coarse-Grained**

Object-oriented technologies such as Java expose their services through individual methods. An individual method is too fine an operation to provide any useful capability at a corporate level. Building a Java program from scratch requires the creation of several fine-grained methods that are then composed into a coarse-grained service that is consumed by either a client or another service.

Businesses and the interfaces that they expose should be coarse-grained. Web services technology provides a natural way of defining coarse-grained services that access the right amount of business logic.

**Ability to be Synchronous or Asynchronous**

Synchronicity refers to the binding of the client to the execution of the service. In synchronous invocations, the client blocks and waits for the service to complete its operation before continuing. Asynchronous operations allow a client to invoke a service and then execute other functions.

Asynchronous clients retrieve their result at a later point in time, while synchronous clients receive their result when the service has completed. Asynchronous capability is a key factor in enabling loosely coupled systems.

**Supports Remote Procedure Calls (RPCs)**

Web services allow clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support.

Component development through Enterprise JavaBeans (EJBs) and .NET Components has increasingly become a part of architectures and enterprise deployments over the past couple of years. Both technologies are distributed and accessible through a variety of RPC mechanisms.

A web service supports RPC by providing services of its own, equivalent to those of a traditional component, or by translating incoming invocations into an invocation of an EJB or a .NET component.

**Supports Document Exchange**

One of the key advantages of XML is its generic way of representing not only data, but also complex documents. These documents can be as simple as representing a current address, or they can be as complex as representing an entire book or Request for Quotation (RFQ). Web services support the transparent exchange of documents to facilitate business integration.

**Web Service Roles**

There are three major roles within the web service architecture:

**Service Provider**

This is the provider of the web service. The service provider implements the service and makes it available on the Internet.

**Service Requestor**

This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.

**Service Registry**

This is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones. It therefore serves as a centralized clearing house for companies and their services.

**Web Service Protocol Stack**

A second option for viewing the web service architecture is to examine the emerging web service protocol stack. The stack is still evolving, but currently has four main layers.

**Service Transport**

This layer is responsible for transporting messages between applications. Currently, this layer includes Hyper Text Transport Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP), and newer protocols such as Blocks Extensible Exchange Protocol (BEEP).

**XML Messaging**

This layer is responsible for encoding messages in a common XML format so that messages can be understood at either end. Currently, this layer includes XML-RPC and SOAP.

**Service Description**

This layer is responsible for describing the public interface to a specific web service. Currently, service description is handled via the Web Service Description Language (WSDL).

**Service Discovery**

This layer is responsible for centralizing services into a common registry and providing easy publish/find functionality. Currently, service discovery is handled via Universal Description, Discovery, and Integration (UDDI).

As web services evolve, additional layers may be added and additional technologies may be added to each layer.

**Few Words about Service Transport**

The bottom of the web service protocol stack is service transport. This layer is responsible for actually transporting XML messages between two computers.

**Hyper Text Transfer Protocol (HTTP)**

Currently, HTTP is the most popular option for service transport. HTTP is simple, stable, and widely deployed. Furthermore, most firewalls allow HTTP traffic. This allows XML-RPC or SOAP messages to masquerade as HTTP messages. This is good if you want to integrate remote applications, but it does raise a number of security concerns.

**Blocks Extensible Exchange Protocol (BEEP)**

This is a promising alternative to HTTP. BEEP is a new Internet Engineering Task Force (IETF) framework for building new protocols. BEEP is layered directly on TCP and includes a number of built-in features, including an initial handshake protocol, authentication, security, and error handling. Using BEEP, one can create new protocols for a variety of applications, including instant messaging, file transfer, content syndication, and network management.

SOAP is not tied to any specific transport protocol. In fact, you can use SOAP via HTTP, SMTP, or FTP. One promising idea is therefore to use SOAP over BEEP.

**Popular SOA Suites**

 Oracle SOA Suite

 IBM SOA Suite

**Most Popular Web Services Frameworks**

 JAX-RPC SI

 Apache AXIS

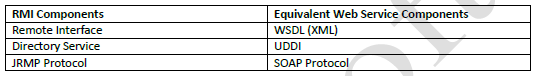
 JAX-WS RI or Metro

 Apache Axis2

 Apache CXF

**How web service solves all the problems of RMI Technology? That means what makes a web service as Language independent, OS Independent, and Firewall free.**

RMI components (Remote interface and implementation, Directory Service and protocol) are language dependent. They are specific to JAVA. RMI uses JRMP Protocol. It is also Language Dependent. Web services uses SOAP protocol and messages. It is plain XML document only. It is Language and OS independent. Web Services uses UDDI (Universal Description, Discovery and Integration) registry to publish and access them. If we use UDDI as a directory service, then we do NOT need to do lookups to get web services reference. Web Services uses WSDL (Web Service Description Language). It is plain xml document only. It is also Language and OS independent.

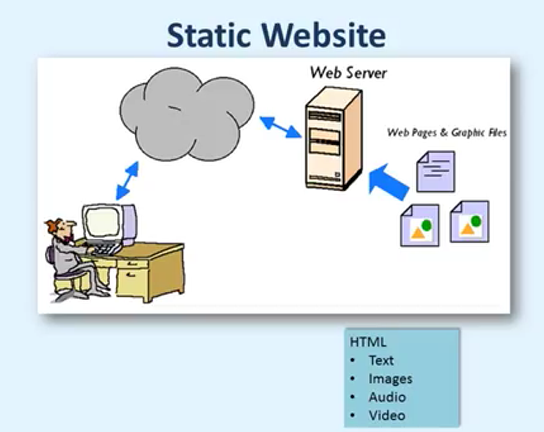


**Web Applications Vs Web Site:**

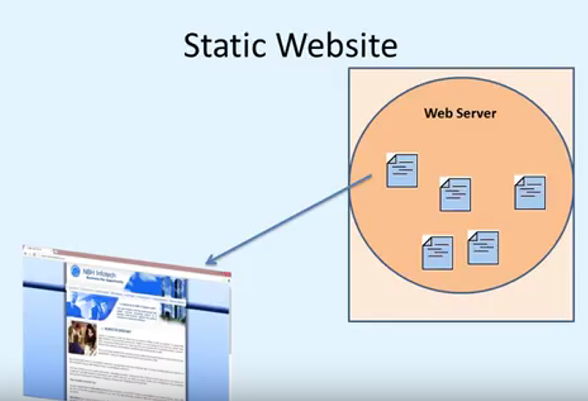
**“Web site is a collection of web pages.”**

**Web Page :**  
Web page is developed using HTML. HTML page may contain text, video, images etc. It’s developed by company for publishing there products, by governments to publish some schemes, etc.

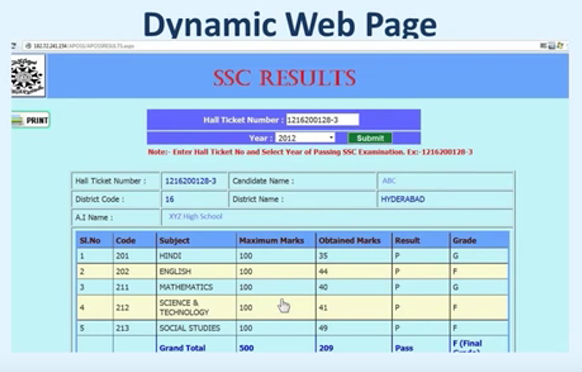
**Static web page:**Static web pages never change. Look and feel and contents never change.  
**How static web page works?**

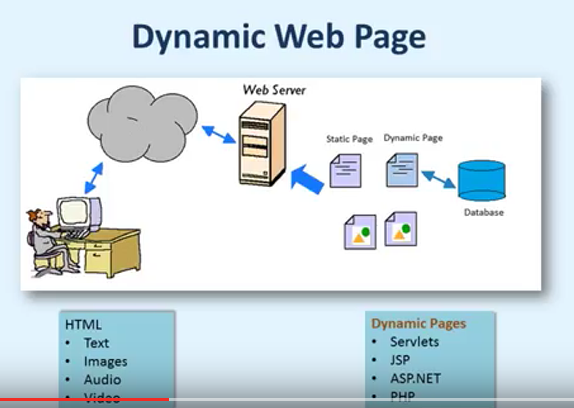


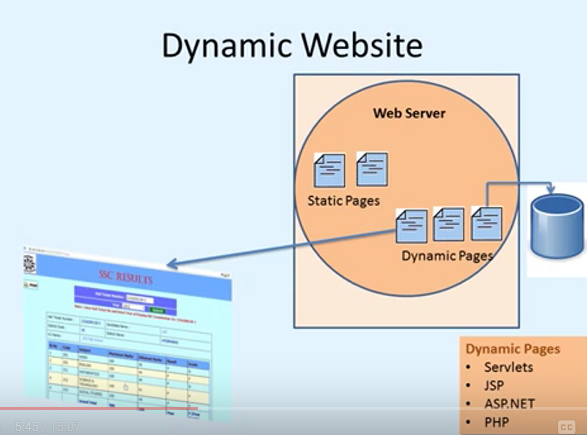
Readymade pages are available and they are loaded directly.

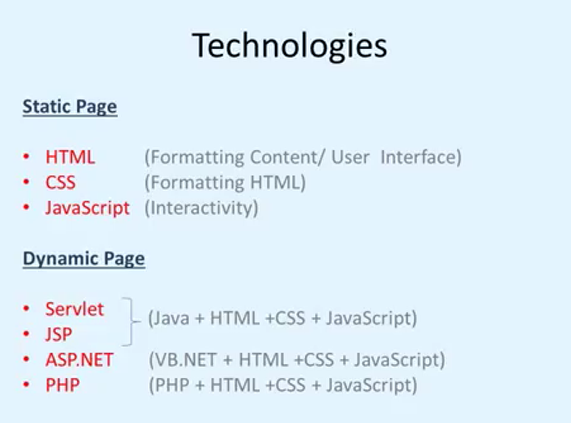


**Dynamic Web pages:**

Her look and feel is same for all the roll numbers we enter as an input. But the marks will be changed every time. So, we can say dynamic web page takes the contents from the user and process it and produce the result. 

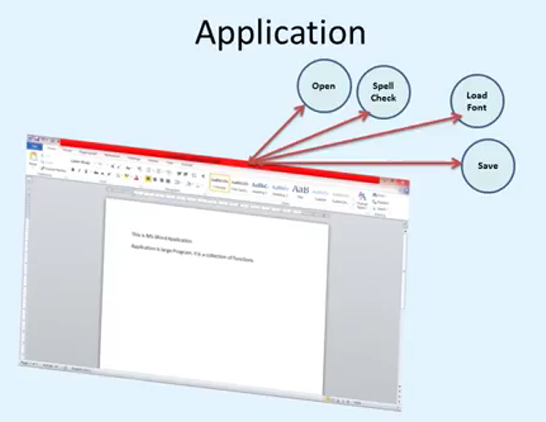






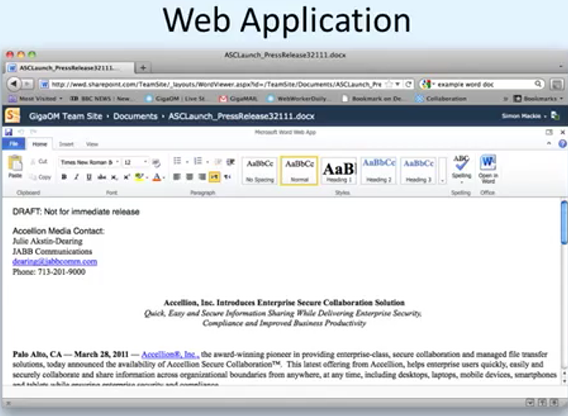
**Application:**

It is the collection of functions. Like MS Word has a function called open, font loader etc., for every activity there will be a function call.



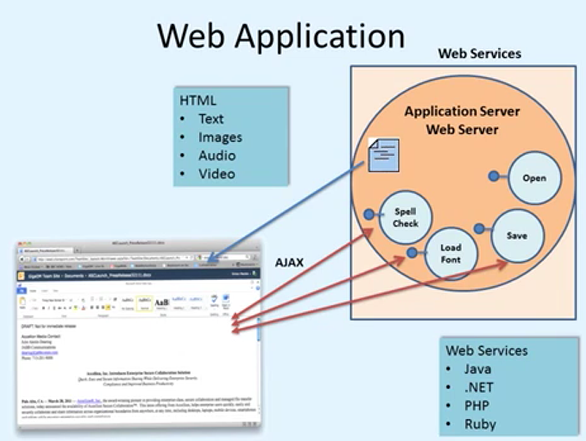
**Example**:

MS Office has a cloud based application called office 365. Here we can see MS word can be loaded in browser window.



Log on to Office 365. The HTML page is loaded in browser. This page is loaded once.

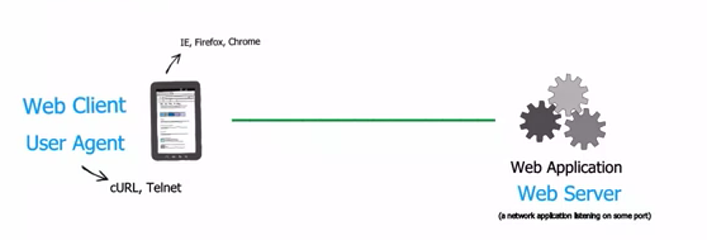
Whatever the activity/functions we do then a corresponding function is called. Functions are present in the web server only. This web page calls those functions from web server. This procedure is similar to cloud computing also.



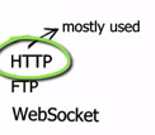
A web application is a piece of software which can be accessed from a browser.

We can call a web application as web server although there are some differences i.e. a web server can host multiple web applications. 

Web server is a network applications running on a machine listening on a port. Browser is also called as web client of user agents.

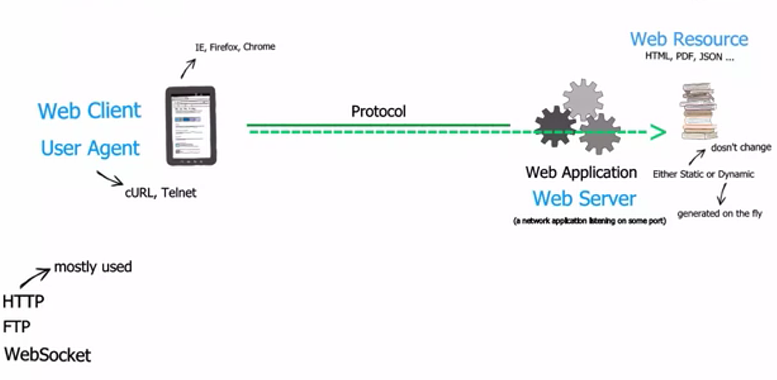


To get the job done the web client communicates to the web server using a protocol called as HTTP.



**Web Resources:**A resource is a document/file like HTML,JSON, PDF, XML etc. which is hosted by a web server.

A web client can access the web resources through the web server.



The resources on the web can be identified by a unique name/string called as URL.

