What is Web Service

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

* is a client server application or application component for communication.
* method of communication between two devices over network.
* is a software system for interoperable machine to machine communication.
* is a collection of standards or protocols for exchanging information between two devices or application.

Let's understand it by the figure given below:



Web Service Components

There are three major web service components.

1. SOAP
2. WSDL
3. UDDI

## SOAP

SOAP is an acronym for Simple Object Access Protocol.

SOAP is a XML-based protocol for accessing web services.

## WSDL

WSDL is an acronym for Web Services Description Language.

WSDL is a xml document containing information about web services such as method name, method parameter and how to access it.

WSDL is a part of UDDI. It acts as a interface between web service applications.

## UDDI

UDDI is an acronym for Universal Description, Discovery and Integration.

UDDI is a XML based framework for describing, discovering and integrating web services.

UDDI is a directory of web service interfaces described by WSDL, containing information about web services.

## Disadvantages of Soap Web Services

**Slow**: SOAP uses XML format that must be parsed to be read. It defines many standards that must be followed while developing the SOAP applications. So it is slow and consumes more bandwidth and resource.

**WSDL dependent**: SOAP uses WSDL and doesn't have any other mechanism to discover the service.

# RESTful Web Services

REST stands for REpresentational State Transfer.

REST is an architectural style not a protocol.

## Advantages of RESTful Web Services

**Fast**: RESTful Web Services are fast because there is no strict specification like SOAP. It consumes less bandwidth and resource.

**Language and Platform independent**: RESTful web services can be written in any programming language and executed in any platform.

**Can use SOAP**: RESTful web services can use SOAP web services as the implementation.

**Permits different data format**: RESTful web service permits different data format such as Plain Text, HTML, XML and JSON.

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| SOAP is a **protocol**. | REST is an **architectural style**. |
| SOAP stands for **Simple Object Access Protocol**. | REST stands for **Representational State Transfer**. |
| SOAP **can't use REST** because it is a protocol. | REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP. |
| SOAP **defines standards**to be strictly followed. | REST does not define too much standards like SOAP. |
| SOAP **requires more bandwidth** and resource than REST. | REST **requires less bandwidth** and resource than SOAP. |
| SOAP **permits XML** data format only. | REST **permits different** data format such as Plain text, HTML, XML, JSON etc. |
| SOAP is **less preferred** than REST. | REST **more preferred** than SOAP. |

# [What's the difference between REST & RESTful](https://stackoverflow.com/questions/1568834/whats-the-difference-between-rest-restful)

REST is an "architectural style" that basically exploits the existing technology and protocols of the Web.

RESTful is typically used to refer to web services implementing such an architecture.

**Q #20) What are HTTP status codes? Enlist few with meaning.**

HTTP status codes basically are the representation of the status of the task that has been performed on the server, with the mode of some codes. Every code has their own meaning.

**Some of the HTTP status codes with their meaning are as follows:**

* **Code 200:** This indicates success.
* **Code 201:** This indicates resource has been successfully created.
* **Code 204:** This indicates that there is no content in the response body.
* **Code 404:** This indicates that there is no method available.

### 3) What are the advantages of web services?

* **Interoperability**: By the help of web services, an application can communicate with other application developed in any language.
* **Reusability**: We can expose the web service so that other applications can use it.
* **Modularity**: By the help of web service, we can create a service for a specific task such as tax calculation etc.

### 15) What is the advantage of XML in web service?

In Web service, an XML is used to tag the data, format the data.

### 17) What is Interoperability in Web services?

Web services facilitate various applications to communicate with 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 communicate with a Java web services and vice versa. Web services are used to make the application platform and technology independent.

## 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'.

**Q-3. Explain the Difference Between HTTP And HTTPS?**

Ans. Following are the key differences between HTTP and HTTPS.

* HTTP stands for Hypertext Transfer Protocol which is a set of rules for passing the information back and forth between clients and web servers. HTTPS refers to a combination of a normal HTTP interaction over an encrypted Secure Sockets Layer (SSL) or Transport Layer Security (TLS) transport mechanism.
* HTTP uses port number 80 whereas, HTTPS uses port number 443.
* HTTP supports mechanism of sending a file to the client asking for it only if any update is there. Whereas, HTTPS encrypts user page requests and decrypts the page response that web server returns.

**What REST stands for?**

REST stands for Representational State Transfer.

**What is REST?**

REST is web standards based architecture and uses HTTP Protocol for data communication. It revolves around resource where every component is a resource and a resource is accessed by a common interface using HTTP standard methods.

In REST architecture, a REST Server simply provides access to resources and REST client accesses and presents the resources. Here each resource is identified by URIs/ global IDs. REST uses various representations to represent a resource like text, JSON and XML. Now a days JSON is the most popular format being used in web services.

**Name some of the commonly used HTTP methods used in REST based architecture?**

**Following well known HTTP methods are commonly used in REST based architecture −**

* **GET** − Provides a read only access to a resource.
* **PUT** − Used method is call when user have to modify/update an existing resource otherwise create operation should happen if Request-URI is a valid resource URI
* **DELETE** − Used to remove a resource.
* **POST** − Used create a new resource.
* **OPTIONS** − Used to get the supported operations on a resource.
* **PATCH** - is a method of modifying resources where the client sends partial data that is to be updated without modifying the entire data.

**What is a Resource in REST?**

REST architecture treats every content as a resource. These resources can be text files, html pages, images, videos or dynamic business data. REST Server simply provides access to resources and REST client accesses and modifies the resources. Here each resource is identified by URIs/ global IDs.

**What are the core components of a HTTP Request?**

A HTTP Request has five major parts −

* **Verb** − Indicate HTTP methods such as GET, POST, DELETE, PUT etc.
* **URI** − Uniform Resource Identifier (URI) to identify the resource on server.
* **HTTP Version** − Indicate HTTP version, for example HTTP v1.1 .
* **Request Header** − Contains metadata for the HTTP Request message as key-value pairs. For example, client (or browser) type, format supported by client, format of message body, cache settings etc.
* **Request Body** − Message content or Resource representation.

**What are the core components of a HTTP response?**

A HTTP Response has four major parts −

* **Status/Response Code** − Indicate Server status for the requested resource. For example 404 means resource not found and 200 means response is ok.
* **HTTP Version** − Indicate HTTP version, for example HTTP v1.1 .
* **Response Header** − Contains metadata for the HTTP Response message as key-value pairs. For example, content length, content type, response date, server type etc.
* **Response Body** − Response message content or Resource representation.

**What is URI?**

URI stands for Uniform Resource Identifier. Each resource in REST architecture is identified by its URI.

**What is purpose of a URI in REST based webservices?**

Purpose of an URI is to locate a resource(s) on the server hosting the web service.

**What is format of a URI in REST architecture?**

A URI is of following format −

<protocol>://<service-name>/<ResourceType>/<ResourceID>

**What is the purpose of HTTP Verb in REST based webservices?**

VERB identifies the operation to be performed on the resource.

**What should be the purpose of OPTIONS method of RESTful web services?**

It should list down the supported operations in a web service and should be read only.

## HTTP Status Codes

1xx: Informational - Request received, continuing process

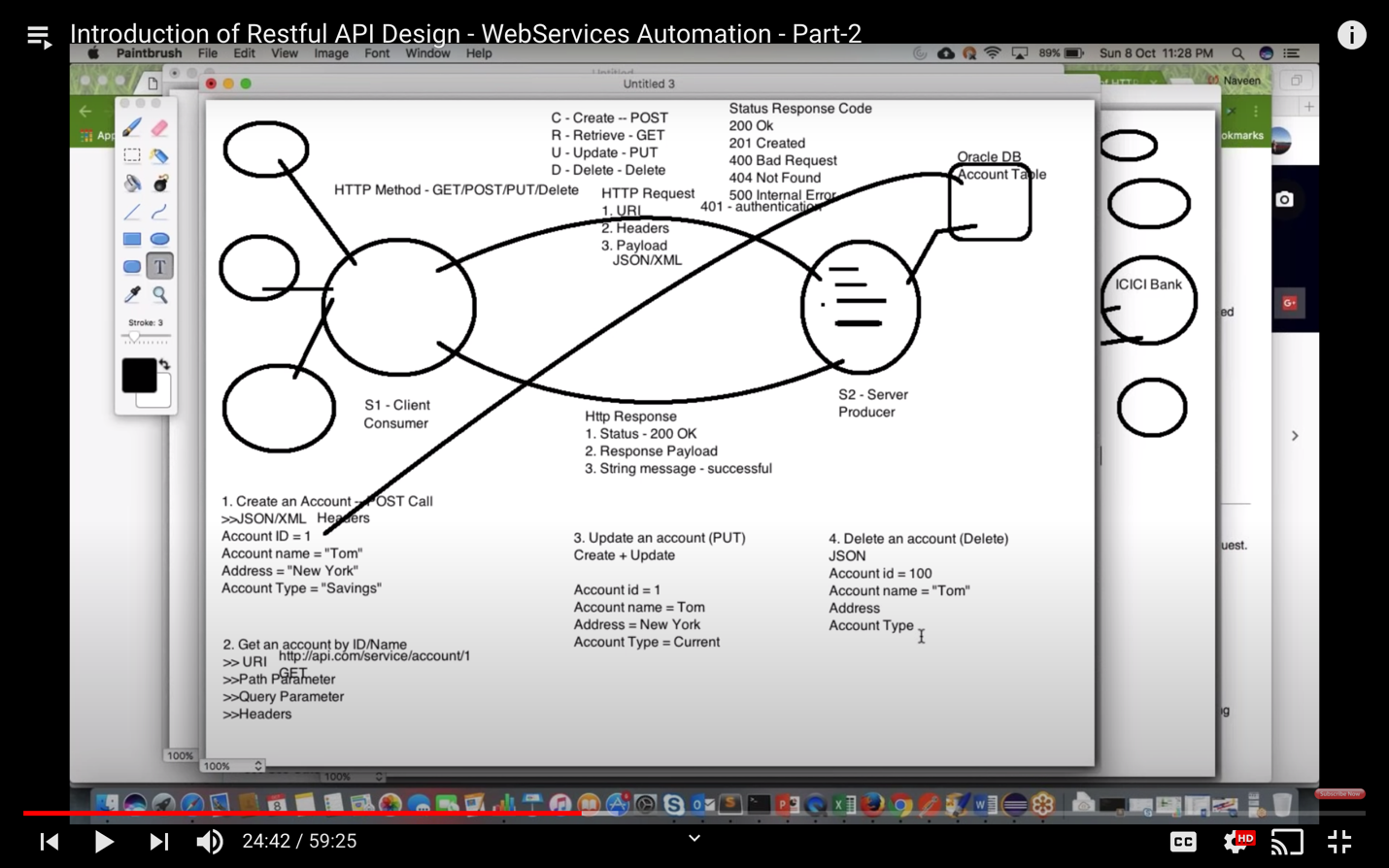
2xx: Success - The action was successfully received, understood, and accepted

3xx: Redirection - Further action must be taken to complete the request

4xx: Client Error - The request contains bad syntax or cannot be fulfilled

5xx: Server Error - The server failed to fulfill an apparently valid request

|  |  |
| --- | --- |
| 200 OK | The request is OK. |
| 201 Created | The request is complete, and a new resource is created. |
| 301 Moved Permanently | The requested page has moved to a new URL . |
| 302 Found | The requested page has moved temporarily to a new url . |
| 400 Bad Request | The server did not understand the request. |
| 401 Unauthorized | The requested page needs a username and a password. |
| 403 Forbidden | Access is forbidden to the requested page. |
| 404 Not Found | The server cannot find the requested page. |
| 500 Internal Server Error | The request was not completed. The server met an unexpected condition. |
| 501 Not Implemented | The request was not completed. The server did not support the functionality required. |
| 502 Bad Gateway | The request was not completed. The server received an invalid response from the upstream server. |



**What is Swagger?**

Swagger is an Interface Description Language for describing RESTful APIs expressed using JSON. Swagger is used together with a set of open-source software tools to design, build, document, and use RESTful web services.

**Performance Testing:**

## Types of Performance Testing

* **Load testing –** checks the application’s ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.
* **Stress testing –** involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.
* **Endurance testing –** is done to make sure the software can handle the expected load over a long period of time.
* **Spike testing –** tests the software’s reaction to sudden large spikes in the load generated by users.
* **Volume testing** – Under Volume Testing large no. of. Data is populated in a database and the overall software system’s behavior is monitored. The objective is to check software application’s performance under varying database volumes.
* **Scalability testing**– The objective of scalability testing is to determine the software application’s effectiveness in “scaling up” to support an increase in user load. It helps plan capacity addition to your software system.

## Common Performance Problems

* **Long Load time –** Load time is normally the initial time it takes an application to start. This should generally be kept to a minimum. While some applications are impossible to make load in under a minute, Load time should be kept under a few seconds if possible.
* **Poor response time –** Response time is the time it takes from when a user inputs data into the application until the application outputs a response to that input. Generally, this should be very quick. Again, if a user has to wait too long, they lose interest.
* **Poor scalability –** A software product suffers from poor scalability when it cannot handle the expected number of users or when it does not accommodate a wide enough range of users. [Load Testing](https://www.guru99.com/load-testing-tutorial.html) should be done to be certain the application can handle the anticipated number of users.
* **Bottlenecking –** Bottlenecks are obstructions in a system which degrade overall system performance. Bottlenecking is when either coding errors or hardware issues cause a decrease of throughput under certain loads. Bottlenecking is often caused by one faulty section of code. The key to fixing a bottlenecking issue is to find the section of code that is causing the slowdown and try to fix it there. Bottlenecking is generally fixed by either fixing poor running processes or adding additional Hardware. Some **common performance bottlenecks** are
  + CPU utilization
  + Memory utilization
  + Network utilization
  + Operating System limitations
  + Disk usage

## Performance Testing Metrics: Parameters Monitored

* **Processor Usage –** an amount of time processor spends executing non-idle threads.
* **Memory use –** amount of physical memory available to processes on a computer.
* **Disk time –**amount of time disk is busy executing a read or write request.
* **Bandwidth –** shows the bits per second used by a network interface.
* **Private bytes –** number of bytes a process has allocated that can’t be shared amongst other processes. These are used to measure memory leaks and usage.
* **Committed memory –** amount of virtual memory used.
* **Memory pages/second –** number of pages written to or read from the disk in order to resolve hard page faults. Hard page faults are when code not from the current working set is called up from elsewhere and retrieved from a disk.
* **Page faults/second –** the overall rate in which fault pages are processed by the processor. This again occurs when a process requires code from outside its working set.
* **CPU interrupts per second –** is the avg. number of hardware interrupts a processor is receiving and processing each second.
* **Disk queue length –** is the avg. no. of read and write requests queued for the selected disk during a sample interval.
* **Network output queue length –** length of the output packet queue in packets. Anything more than two means a delay and bottlenecking needs to be stopped.
* **Network bytes total per second –** rate which bytes are sent and received on the interface including framing characters.
* **Response time –** time from when a user enters a request until the first character of the response is received.
* **Throughput –** rate a computer or network receives requests per second.
* **Amount of connection pooling –** the number of user requests that are met by pooled connections. The more requests met by connections in the pool, the better the performance will be.
* **Maximum active sessions –** the maximum number of sessions that can be active at once.
* **Hit ratios –** This has to do with the number of[SQL](https://www.guru99.com/sql.html)statements that are handled by cached data instead of expensive I/O operations. This is a good place to start for solving bottlenecking issues.
* **Hits per second –** the no. of hits on a web server during each second of a load test.
* **Rollback segment –** the amount of data that can rollback at any point in time.
* **Database locks –** locking of tables and databases needs to be monitored and carefully tuned.
* **Top waits –** are monitored to determine what wait times can be cut down when dealing with the how fast data is retrieved from memory
* **Thread counts –** An applications health can be measured by the no. of threads that are running and currently active.
* **Garbage collection –** It has to do with returning unused memory back to the system. Garbage collection needs to be monitored for efficiency.

## Example Performance Test Cases

* Verify response time is not more than 4 secs when 1000 users access the website simultaneously.
* Verify response time of the Application Under Load is within an acceptable range when the network connectivity is slow
* Check the maximum number of users that the application can handle before it crashes.
* Check database execution time when 500 records are read/written simultaneously.
* Check CPU and memory usage of the application and the database server under peak load conditions
* Verify response time of the application under low, normal, moderate and heavy load conditions.

**How to Create EC2 Instance in AWS: Step by Step Tutorial**

<https://www.guru99.com/creating-amazon-ec2-instance.html>