**Web Services** allows different applications to talk to each other and share data and services among themselves. Other applications can also **use** the services of the **web services**. For example VB or .NET application can talk to **java web services** and vice versa.

**Why Use WebServices?**

1. Exposing the Existing Function on the network:
2. Inter-Operatorability : 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.
3. Standardize Protocol.
4. Low Cost communnication.

**Difference b/w soap and rest**

|  |  |
| --- | --- |
| **Soap** | **Rest** |
| Soap is 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’s a protocol | Rest can use soap because it can use any protocol like http, soap |
| JAX-WS java API for soap | JAX-RS java API for rest |
| Soap defines standard to be strictly followed | REST does not define too much standards like SOAP. |
| SOAP **uses services interfaces to expose the business logic.** | REST **uses URI to expose business logic**. |
| SOAP **permits XML** data format only. | Support multiple data format.REST **permits different** data format such as Plain text, HTML, XML, JSON etc. |
| SOAP is **less preferred** than REST. | REST **more preferred** than SOAP. |
| Support stateless and statefull/conversational operation | Support stateless conversational operation |
| Uses Http Post | Uses Http Verbs(GET/POST/DELETE..) |
|  |  |

## Get vs. Post

There are many differences between the Get and Post request. Let's see these differences:

|  |  |
| --- | --- |
| **GET** | **POST** |
| 1) In case of Get request, only **limited amount of data**can be sent because data is sent in header. | In case of post request, **large amount of data**can be sent because data is sent in body. |
| 2) Get request is **not secured**because data is exposed in URL bar. | Post request is **secured**because data is not exposed in URL bar. |
| 3) Get request **can be bookmarked.** | Post request **cannot be bookmarked.** |
| 4) Get request is **idempotent**. It means second request will be ignored until response of first request is delivered | Post request is **non-idempotent.** |
| 5) Get request is **more efficient**and used more than Post. | Post request is **less efficient**and used less than get. |

<https://www.w3schools.com/tags/ref_httpmethods.asp>

**HTTP Methods:** GET/POST/PUT/HEAD/DELETE/PATCH/OPTIONS

## The PUT Method

**PUT is used to send data to a server to create/update a resource.**

**Note:** The difference between POST and PUT is that PUT requests are idempotent. That is, calling the same PUT request multiple times will always produce the same result. In contrast, calling a POST request repeatedly have side effects of creating the same resource multiple times.

**The POST Method**

**POST is used to send data to a server to create/update a resource.**

The data sent to the server with POST is stored in the request body of the HTTP request:

Exam:

POST /test/demo\_form.php HTTP/1.1  
Host: w3schools.com  
name1=value1&name2=value2

* POST is one of the most common HTTP methods.
* POST requests are never cached
* POST requests do not remain in the browser history
* POST requests cannot be bookmarked.
* POST requests have no restrictions on data length.

## The HEAD Method

* **HEAD is almost identical to GET, but without the response body.**
* In other words, if GET /users returns a list of users, then HEAD /users will make the same request but will not return the list of users.
* HEAD requests are useful for checking what a GET request will return before actually making a GET request - like before downloading a large file or response body.

## The DELETE Method

**The DELETE method deletes the specified resource.**

DELETE is pretty easy to understand. It is used to **delete** a resource identified by a URI.

On successful deletion, return HTTP status 200 (OK) along with a response body, perhaps the representation of the deleted item (often demands too much bandwidth), or a wrapped response (see Return Values below). Either that or return HTTP status 204 (NO CONTENT) with no response body. In other words, a 204 status with no body, or the JSEND-style response and HTTP status 200 are the recommended responses.

## The OPTIONS Method

**The OPTIONS method describes the communication options for the target resource.**

**Differences**

1. **POST**
   * If the client sends data without any identifier, then we will store the data and assign/generate a new identifier.
   * If the client again sends the **same** data without any identifier, then we will store the data and assign/generate a new identifier.
   * **Note**: Duplication is allowed here.

The POST verb is mostly utilized to **create** new resources. In particular, it's used to create subordinate resources. That is, subordinate to some other (e.g. parent) resource.

On successful creation, return HTTP status 201, returning a Location header with a link to the newly-created resource with the 201 HTTP status.

**Checking with Fiddler or PostMan:** We can use Fiddler for checking the response. Open Fiddler and select the Compose tab. Specify the verb and url as shown below and click Execute to check the response.

**Verb:** POST

**url:** <http://jsonplaceholder.typicode.com/posts/>

**Request Body:**

data: { title: 'foo', body: 'bar', userId: 1000, Id : 1000 }

**Response:** You would receive the response code as 201.

If we want to check the inserted record with Id = 1000 change the verb to Get and use the same url and click Execute.

**PUT**: If the client sends data with an identifier, then we will check whether that identifier exists. If the identifier exists, we will update the resource with the data, else we will create a resource with the data and assign/generate a new identifier.

If I had to change my firstname then send PUT request for Update:

{ "first": "Nazmul", "last": "hasan" } So, here in order to update the first name we need to send all the parameters of the data again.

**PATCH:** If the client sends data with an identifier, then we will check whether that identifier exists. If the identifier exists, we will update the resource with the data, else we will throw an exception.

Patch request says that we would only send the data that we need to modify without modifying or effecting other parts of the data. Ex: if we need to update only the first name, we pass only the first name.

**Note**: On the **PUT** method, we are not throwing an exception if an identifier is not found. But in the **PATCH** method, we are throwing an exception if the identifier is not found.

**-------------------------------------------------------------------------------------------------------------**

**Http Error Status:**

|  |  |
| --- | --- |
| 400: Bad Request | A HyperText Transfer Protocol Bad Request 400 indicates a client error. When the client sends an invalid request to the server, the server issues an HTTP status code 400. |
| 401 : Unauthorized | The server can't process the request due to client-side errors.  The 401 status code indicates that the HTTP request has not been applied because it lacks valid authentication credentials (usually username and password) for the target resource. |
| 404 Not Found | The server can not find the requested resource. |
| 403 Forbidden | Unauthorized request. The client does not have access rights to the content. Unlike 401, the client’s identity is known to the server. |
| **402 Payment Required (Experimental)** | Reserved for future use. It is aimed for using in the digital payment systems. |
| 405 Method Not Allowed | The request HTTP method is known by the server but has been disabled and cannot be used for that resource. |
| **200 OK** | Indicates that the request has succeeded. |
| **204 No Content** | The server has fulfilled the request but does not need to return a response body. The server may return the updated meta information. |
| 201 Created | Indicates that the request has succeeded and a new resource has been created as a result. |
| 500: Internal Server Error | This is a "catch-all" status for unexpected errors. |
| HTTP status 504 - Gateway timeout | This means, that the server has not responded within the specified time period. |
| 502 Bad Gateway |  |
|  |  |
|  |  |

**https://hetmanrecovery.com/recovery\_news/the-most-common-online-errors-and-how-to-fix-them.htm#:~:text=The%20%C2%AB400%C2%BB%20error%20occurs%20when%20the%20server%20cannot,using%20a%20website%20%C2%ABURL%20address%C2%BB%20that%20doesn%E2%80%99t%20exist.**