***WEB SERVICES***

# **TYPES OF WEB SERVICES Communication Protocols:-**

1. ***SOAP( Simple object access protocol)***

* It is a xml based protocol for accessing web services
* It is platform independent and language independent.
* SOAP defines its own security.
* SOAP is secure than rest API.

Dis Advantages:-

* SOAP uses XMl Format, Must be parse to read.
* SOAP Supports only XML**.**

1. ***REST FUL***

**REST: -** **Representational State Transfer**

**API: - Application Programing Interface**

What is REST API:-

* REST is a Architectural style
* A REST API defines a set of functions to process requests and responses via HTTP protocol.
* A RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.

https://dzone.com/articles/differences-in-performance-apis-amp-more

# ***What is HTTP?***

* The Hypertext Transfer Protocol (HTTP) designed to enable communications between clients and Web servers.
* HTTP works as a request-response protocol between a client and server.
* A web browser may be the client, and an application on a computer that hosts a web site may be the server.
* Example: A client (browser) submits an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may contain the requested content.

**D/B Web Application and Web Services**

**Web Application**

A web application is a software application that a user runs in the web browser. It have the following features.

* It has a User Interface
* Runs in Client - Server environment
* Client runs it with the help of a web browser
* Server process the data based on client request and provide response

**Web Service**

A web service is an Application Program Interface (API) that runs on the server, which provide data to the client over http through a standardized messaging system. (XML, JSON, etc...). Web services are further classified into [SOAP](http://searchsoa.techtarget.com/definition/SOAP) and [REST](http://rest.elkstein.org/). In the present day scenario most services prefer [REST over SOAP](http://spf13.com/post/soap-vs-rest).

***Example***

I'll explain a simple REST call using a GET request to the [github api](https://developer.github.com/v3/" \t "_blank). You can access the details  of a github user with the following request

# ***HTTP Methods***

* **GET**
* **POST**
* **PUT**
* **HEAD**
* **DELETE**
* **PATCH**

# ***GET Method:-***

**GET used to request data from a specified resource.**

**Get is used for search records**

**GET is one of the most common HTTP methods.**

Note that the query string (name/value pairs) is sent in the URL of a GET request:

/test/demo\_form.php?name1=value1&name2=value2

**Some other notes on GET requests:**

* GET requests can be cached
* GET requests remain in the browser history
* GET requests can be bookmarked
* GET requests should never be used when dealing with sensitive data
* GET requests have length restrictions
* GET requests is only used to request data (not modify)

# ***The POST Method:-***

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

A POST request used to send data to the server, for example, customer information, file upload, etc. using HTML forms.

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

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

**POST is one of the most common HTTP methods.**

**Some other notes on POST requests:**

* **To Add new records**
* 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 PUT Method:-***

* **To Add and Update**

Replaces all the current representations of the target resource with the uploaded content.

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

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 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 PATCH Method:-***

**Patch is for Customized updates**

**Exp:- {name:’sriram’}**

**Put is for full updates**

**Exp:- {**

**Name:’sriram’**

**Dep:’IT’**

**}**

Update only the specified fields of an entity at a URI. A PATCH request is neither safe nor idempotent (RFC 5789). That's because a PATCH operation cannot ensure the entire resource has been updated.

PATCH /addresses/1

# The DELETE Method

Removes all the current representations of the target resource given by URI.

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

# **D/B POST AND PUT**

* Post used to create
* Post is a **NON IDEMPOTENT**
* Put is a  **IDEMPOTENT**
* Put used to create and update
* 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

**In post identification number generated by server**

**In put server knows the identification number**

## IDEMPOTENT:- Request and response both are same

## NON IDEMPOTENT:- Request and response both are not same

## 

# **HTTP Status Messages**

* Status codes indicate the result of the HTTP request.
* When a browser requests a service from a web server, an error might occur.

1XX – informational 🡺 It means the request has been received and the process is continuing.

2XX – success 🡺 It means the action was successfully received, understood, and accepted.

3XX – redirection 🡺 It means further action must be taken in order to complete the request.

4XX - client error 🡺 It means the request contains incorrect syntax or cannot be fulfilled.

5XX - server error 🡺 It means the server failed to fulfill an apparently valid request.

1xx: Information

|  |  |
| --- | --- |
| **Message** | **Description** |
| 100 Continue | The server has received only a part of the request, but as long as it has not been rejected, the client should continue with the request. |
| 101 Switching Protocols | The server switches protocol. |

2xx: Successful

|  |  |
| --- | --- |
| **Message** | **Description** |
| 200 OK | The request is OK. |
| 201 Created | The request is complete, and a new resource created. |
| 202 Accepted | The request is accepted for processing, but the processing is not complete. |
| 203 Non-authoritative Information | The information in the entity header is from a local or third-party copy, not from the original server. |
| 204 No Content | A status code and a header are given in the response, but there is no entity-body in the reply. |
| 205 Reset Content | The browser should clear the form used for this transaction for additional input. |
| 206 Partial Content | The server is returning partial data of the size requested. Used in response to a request specifying a *Range* header. The server must specify the range included in the response with the *Content-Range* header. |

3xx: Redirection

|  |  |
| --- | --- |
| **Message** | **Description** |
| 300 Multiple Choices | A link list. The user can select a link and go to that location. Maximum five addresses  . |
| 301 Moved Permanently | The requested page has moved to a new url . |
| 302 Found | The requested page has moved temporarily to a new url . |
| 303 See Other | The requested page can be found under a different url . |
| 304 Not Modified | This is the response code to an *If-Modified-Since* or *If-None-Match* header, where the URL has not been modified since the specified date. |
| 305 Use Proxy | The requested URL must be accessed through the proxy mentioned in the *Location* header. |
| 306 *Unused* | This code was used in a previous version. It is no longer used, but the code is reserved. |
| 307 Temporary Redirect | The requested page has moved temporarily to a new url. |

4xx: Client Error

|  |  |
| --- | --- |
| **Message** | **Description** |
| 400 Bad Request | The server did not understand the request. |
| 401 Unauthorized | The requested page needs a username and a password. |
| 402 Payment Required | *You can not use this code yet*. |
| 403 Forbidden | Access is forbidden to the requested page. |
| 404 Not Found | The server can not find the requested page. |
| 405 Method Not Allowed | The method specified in the request is not allowed. |
| 406 Not Acceptable | The server can only generate a response that is not accepted by the client. |
| 407 Proxy Authentication Required | You must authenticate with a proxy server before this request can be served. |
| 408 Request Timeout | The request took longer than the server was prepared to wait. |
| 409 Conflict | The request could not be completed because of a conflict. |
| 410 Gone | The requested page is no longer available . |
| 411 Length Required | The "Content-Length" is not defined. The server will not accept the request without it . |
| 412 Precondition Failed | The pre condition given in the request evaluated to false by the server. |
| 413 Request Entity Too Large | The server will not accept the request, because the request entity is too large. |
| 414 Request-url Too Long | The server will not accept the request, because the url is too long. Occurs when you convert a "post" request to a "get" request with a long query information . |
| 415 Unsupported Media Type | The server will not accept the request, because the mediatype is not supported . |
| 416 Requested Range Not Satisfiable | The requested byte range is not available and is out of bounds. |
| 417 Expectation Failed | The expectation given in an Expect request-header field could not be met by this server. |

5xx: Server Error

|  |  |
| --- | --- |
| **Message** | **Description** |
| 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. |
| 503 Service Unavailable | The request was not completed. The server is temporarily overloading or down. |
| 504 Gateway Timeout | The gateway has timed out. |
| 505 HTTP Version Not Supported | The server does not support the "http protocol" version. |