**1. HTTP Protocol and Methods**

Hypertext Transfer Protocol (HTTP) is a Protocol used to transfer the data over the web.

HTTP work as request-response protocol between client and server.

**1.1 HTTP METHODS**

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

**GET**

HTTP GET Request is a request for getting a representation or information of the resource. This request does not change any resource state on the server, because of that we can refer this method as a safe method.

GET is a idempotent in nature,which means that multiple identical requests produce the same result on every request.

If we have a URL to a resource and don’t know anything more, we can alway make a GET request and get a representation in return.

**POST**

The POST method is used to create a new element to a resource . In other words, POST request is used to send data to server for creating or updating a resource.

POST is no-idempotent in nature ,which means that multiple identical requests produce different result on every request.

RFC 2616, the HTTP specification, has this to say about POST:

POST is designed to allow a uniform method to cover the following functions:

* Annotation of existing resources;
* Posting a message to a bulletin board, newsgroup, mailing list, or similar group of articles;
* Providing a block of data, such as the result of submitting a form, to a data-handling process;
* Extending a database through an append operation.

**PUT**

A PUT request is a request to modify resource state. PUT request update existing resource.

PUT request is a idempotent in nature,which means that multiple identical requests produce the same result on every request.

**HEAD**:

HEAD is a safe method and almost identical to the GET method.

HEAD is a lightweight version of get method.The server treat HEAD request exactly same as a GET request,but not supposed to send a an entity-body -only the HTTP status code and the headers.

**DELETE**

DELETE request is used when we want to delete some resources.DELETE request is destroy the resource and never refer to it again. DELETE is obviously not a safe method.

The DELETE method has another useful property: it’s idempotent.

**PATCH**

PATCH request is used to make partial update on a resource.The difference between PUT and PATCH request is PUT method is used when we want to change the entire representation of resource and PATCH is used only when we want to make some changes in representation don’t want to change entire resource.

**OPTIONS**

OPTIONS is a primitive discovery mechanism for HTTP. The response to an OPTIONS request contains the HTTP Allow header, which laysout which HTTP methods the resource supports.

**1.2 STATUS CODE**

Status codes is response for our http methods.Each Status-Code is described below :

### **1.2.1 Informational 1xx**

This class of status code indicates the request has been received and the process is continuing.

|  |  |  |
| --- | --- | --- |
| **STATUS CODE** | **MESSAGE** | **DETAIL** |
| 100 | Continue | The client SHOULD continue with its request. This response is used to inform the client that the initial part of the request has been received and has not yet been rejected by the server. The client SHOULD continue by sending the remainder of the request |
| 101 | Switching Protocols | The server will switch protocols to those defined by the response's Upgrade header field immediately after the empty line which terminates the 101 response. |

### **1.2.2 Successful 2xx**

This class of status code indicates that the client's request was successfully received, understood, and accepted.

|  |  |  |
| --- | --- | --- |
| **STATUS CODE** | **MESSAGE** | **DETAILS** |
| 200 | OK | The request has succeeded |
| 201 | Created | The request has been fulfilled and resulted in a new resource being created. |
| 202 | Accepted | The request has been accepted for processing, but the processing has not been completed. |
| 203 | Non-Authoritative Information | The returned meta information in the entity-header is not the definitive set as available from the origin server, but is gathered from a local or a third-party copy. |
| 204 | No Content | The server has fulfilled the request but does not need to return an entity-body, and might want to return updated meta information. |
| 205 | Reset Content | The server has fulfilled the request and the user agent SHOULD reset the document view which caused the request to be sent. |
| 206 | Partial Content | The server has fulfilled the partial GET request for the resource. |

### **1.2.3 Redirection 3xx**

This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request.

|  |  |  |
| --- | --- | --- |
| **STATUS CODE** | **MESSAGE** | **DETAILS** |
| 300 | Multiple Choices | The response SHOULD include an entity containing a list of resource characteristics and location(s) from which the user or user agent can choose the one most appropriate. |
| 301 | Moved Permanently | The requested resource has been assigned a new permanent URI and any future references to this resource SHOULD use one of the returned URIs. |
| 302 | Found | The requested resource resides temporarily under a different URI.Since the redirection might be altered on occasion, the client SHOULD continue to use the Request-URI for future requests. |
| 303 | See Other | The response to the request can be found under a different URI and SHOULD be retrieved using a GET method on that resource. |
| 304 | Not Modified | If the client has performed a conditional GET request and access is allowed, but the document has not been modified, the server SHOULD respond with this status code. |
| 305 | Use Proxy | The requested resource MUST be accessed through the proxy given by the Location field. |
| 307 | Temporary Redirect | The requested resource resides temporarily under a different URI.Since the redirection MAY be altered on occasion, the client SHOULD continue to use the Request-URI for future requests. |

### **1.2.4 Client Error 4xx**

The 4xx class of status code is intended for cases in which the client seems to have erred.

|  |  |  |
| --- | --- | --- |
| **STATUS CODE** | **MESSAGE** | **DETAILS** |
| 400 | Bad Request | The request could not be understood by the server due to malformed syntax. |
| 401 | Unauthorized | The request requires user authentication. |
| 402 | Payment Required | This code is reserved for future use. |
| 403 | Forbidden | The server understood the request, but is refusing to fulfill it. |
| 404 | Not Found | The server has not found anything matching the Request-URI. |
| 405 | Method Not Allowed | The method specified in the Request-Line is not allowed for the resource identified by the Request-URI. |
| 406 | Not Acceptable | The resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headers sent in the request. |
| 407 | Proxy Authentication Required | This code is similar to 401 (Unauthorized), but indicates that the client must first authenticate itself with the proxy |
| 408 | Request Timeout | The client did not produce a request within the time that the server was prepared to wait. |
| 409 | Conflict | The request could not be completed due to a conflict with the current  state of the resource. |
| 410 | Gone | The requested resource is no longer available at the server and no forwarding address is known. |
| 411 | Length Required | The server refuses to accept the request without a defined Content- Length. |
| 412 | Precondition Failed | The precondition given in one or more of the request-header fields evaluated to false when it was tested on the server. |
| 413 | Request Entity Too Large | The server is refusing to process a request because the request entity is larger than the server is willing or able to process. |
| 414 | Request-URI Too Long | The server is refusing to service the request because the Request-URI is longer than the server is willing to interpret. |
| 415 | Unsupported Media Type | The server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method. |
| 416 | Requested Range Not Satisfiable | A server SHOULD return a response with this status code if a request included a Range request-header field, and none of the range-specifier values in this field overlap the current extent of the selected resource, and the request did not include an If-Range request-header field. |
| 417 | Expectation Failed | The expectation given in an Expect request-header field |

### **1.2.5 Server Error 5xx**

Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has erred or is incapable of performing the request.

|  |  |  |
| --- | --- | --- |
| **STATUS CODE** | **MESSAGE** | **DETAILS** |
| 500 | Internal Server Error | The server encountered an unexpected condition which prevented it from fulfilling the request. |
| 501 | Not Implemented | The server does not support the functionality required to fulfill the request. |
| 502 | Bad Gateway | The server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfill the request. |
| 503 | Service Unavailable | The server is currently unable to handle the request due to a temporary overloading or maintenance of the server. |
| 504 | Gateway Timeout | The server, while acting as a gateway or proxy, did not receive a timely response from the upstream server specified by the URI (e.g. HTTP, FTP, LDAP) or some other auxiliary server (e.g. DNS) it needed to access in attempting to complete the request. |
| 505 | HTTP Version Not Supported | The server does not support, or refuses to support, the HTTP protocol version that was used in the request message. |

**2. Representational State Transfer( REST)**

**2.1 SOAP API(simple object Access protocol)**

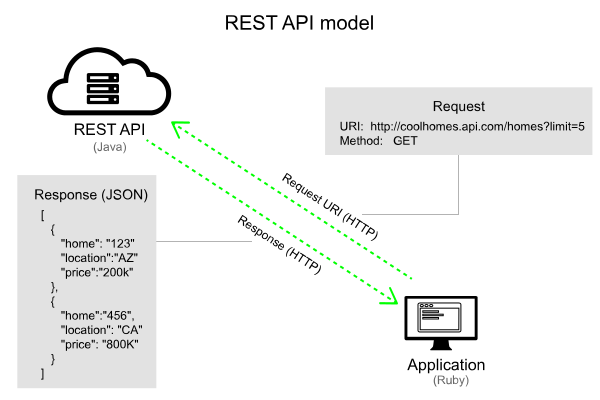
SOAP API is predecessor to the REST API. SOAP is a standardized protocol that requires XML as the message format for requests and responses. As a standardized protocol, the message format is usually defined through something called a WSDL (Web Services Description Language) file.

The WSDL file defines the allowed elements and attributes in the message exchanges. The WSDL file is machine readable and used by the servers interacting with each other to facilitate the communication.SOAP messages are enclosed in an “envelope” that includes a header and body, using a specific XML schema and namespace.

**2.2 What is REST**

Representational State Transfer (REST) is a style of architecture based on a set of principles that describe how networked resources are defined and addressed.These principles were first described in 2000 by Roy Fielding as part of his doctoral dissertation. REST is an alternative to SOAP and JSON.

The following diagram shows the general model of a REST API



**2.3 Architectural Constraints**

REST defines 6 architectural constraints which make any web service – a true RESTful API.

1. **Uniform interface**

The uniform interface constraint is fundamental to the design of any REST service. It simplifies and decouples the architecture, which enables each part to evolve independently.

The four constraints for this uniform interface are:

1. Resource identification in requests
2. Resource manipulation through representations
3. Self-descriptive messages
4. Hypermedia as the engine of application state

### **Client–server architecture**

The principle behind the client–server constraints is the separation of concerns. Separating the user interface concerns from the data storage concerns improves the portability of the user interface across multiple platforms. It also improves scalability by simplifying the server components. Perhaps most significant to the Web, however, is that the separation allows the components to evolve independently, thus supporting the Internet-scale requirement of multiple organizational domains

### **Statelessness**

The client–server communication is constrained by no client context being stored on the server between requests. Each request from any client contains all the information necessary to service the request, and session state is held in the client. The session state can be transferred by the server to another service such as a database to maintain a persistent state for a period and allow authentication. The client begins sending requests when it is ready to make the transition to a new state. While one or more requests are outstanding, the client is considered to be *in transition*. The representation of each application state contains links that can be used the next time the client chooses to initiate a new state-transition.

### **Cacheability**

on the World Wide Web, clients and intermediaries can cache responses. Responses must therefore, implicitly or explicitly, define themselves as cacheable or not to prevent clients from getting stale or inappropriate data in response to further requests. Well-managed caching partially or completely eliminates some client–server interactions, further improving scalability and performance.

### **Layered system**

A client cannot ordinarily tell whether it is connected directly to the end server, or to an intermediary along the way. Intermediary servers can improve system scalability by enabling load balancing and by providing shared caches. They can also enforce security policies.