llApi/restful webservices/ postman/restassured

**What is rest assured ?**

It is a Java lib(api) for testing restful webservices.

To test XML & JSON based web services.

Supports GET, POST, PUT, PATCH, DELETE, OPTIONS, HEAD etc.

Can be integrated with testing frameworks like JUnit, TestNG

Rest assured is implemented in Groovy

**Client:** something which requesting a resource.

**Server:** something which is holding a resource.

**What is an API?**

An API (Application Programming Interface) is a software intermediary that enables two applications to communicate with each other.

In an API testing interview, you could be asked to give some API examples, here are the well-known ones: Google Maps API, Amazon Advertising API, Twitter API, YouTube API, etc.

### What are some architectural styles for creating a Web API?

* HTTP for client-server communication
* XML/JSON as formatting language
* Simple URI as the address for the services
* Stateless communication

### What is API Testing?

[API testing](https://www.katalon.com/resources-center/tutorials/introduction-api-testing/) is a kind of software testing which determines if the developed APIs meet expectations regarding the functionality, reliability, performance, and security of the application.

### What are the advantages of API Testing?

**Test for Core Functionality:**API testing provides access to the application without a user interface.

***Time Effective:*** API testing usually is less time consuming than functional GUI testing.

### Some common protocols used in API testing?

Many protocols are now available to be used in API testing, such as JMS, REST, HTTP, UDDI and SOAP.

**What are the common API testing types?**

While there are certainly specialty tests, and no list can be asked to be comprehensive in this realm, most tests fit broadly into these following nine categories that you should remember before attending in an API testing interview.

1. Validation Testing
2. Functional Testing
3. Load testing
4. Runtime/ Error Detection
5. Security testing
6. Penetration testing
7. Fuzz testing
8. Interoperability and WS Compliance testing

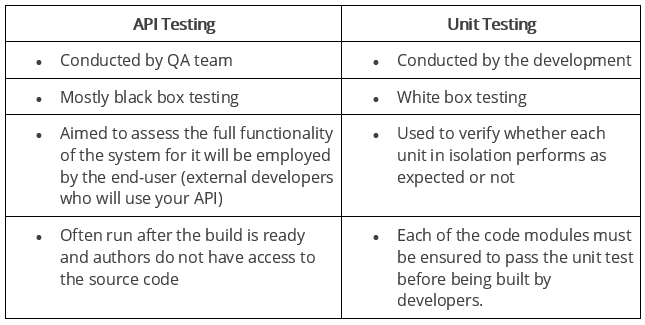
### What must be checked when performing API testing?

During the API testing process, a request is raised to the API with the known data. This way you can analyze the validation response. While testing an API, you should consider:

* **Accuracy of data(correctness of data)**
* **HTTP status codes**
* Data type
* **Validations**
* **Missing functionalities**
* **Authorization checks**
* Implementation of response timeout
* Non-functional testing like **performance and security testing**

### What are diff

### erences between API Testing and Unit Testing?



|  |  |
| --- | --- |
| In API testing there is a wide scope of testing.  API testing is done by the testers. The whole purpose of API testing is end to end testing of the functionality. | In Unit testing there is a limited scope of testing we can test only the basic  functionality.  Unit testing is done by the developer. In unit testing every functionality is  separately tested. |

**What are differences between API Testing and UI Testing?**

* API enables communication between two separate software systems. A software system implementing an API contains functions or subroutines that can be executed by another software system.
* On the other hand, UI ( User Interface) testing refers to testing graphical interface such as how users interact with the applications, testing application elements like fonts, images, layouts etc. UI testing basically focuses on look and feel of an application.

**What are the testing methods that come under API testing?**

One of the most common Web API testing interview questions is about the testing methods. They are:

* Unit testing and **Functional** testing
* Load testing to test the **performance** under load
* Usability and Reliability testing to get consistent results
* **Security** and Penetration testing to validate all types of authentication
* **Automation testing** to create and run scripts that require regular API calls
* End to end Integration and Web UI testing

**Why is API testing considered as the most suitable form for Automation testing?**

API testing is now preferred over GUI testing and is considered as most suitable because:

* It verifies all the functional paths of the system under test very effectively.
* It provides the most stable interface.
* It is easier to maintain and provides fast feedback.

**What are common API errors that often founded?**

Not only API fundamental questions, the interviewer also determine your knowledge and experience by asking about the API errors in a Web API testing interview. So the most common ones are:

* Authentication errors
* API not secure
* Parameter validation errors

**What kinds of bugs that API testing would often find?**

* Missing functionality
* Duplicate functionality error
* Validation errors
* Stress
* Reliability
* Security
* Performance

### What is API documentation?

The API documentation is a complete, accurate technical writing giving instructions on how to effectively use and integrate with an API. It is a compact reference manual that has all the information needed to work with the API, and helps you answer all the API testing questions with details on functions, classes, return types, arguments, and also examples and tutorials.

**What are API documentation templates that are commonly used?**

There are several available API documentation templates help to make the entire process simple and straightforward, which could be  answered in your API testing interview, such as:

* Swagger
* Miredot
* Slate
* FlatDoc
* API blueprint
* RestDoc
* Web service API specification

**What is Rest API**

Rest stands for Representational State Transfer

REST suggests creating an object of the data requested by the client and sending the values of the object in response to the user. For example, if the user is requesting for a movie in Bangalore at a certain place and time, then you can create an object on the server-side.

So, over here, you have an object and you are sending the state of an object(**in form of JSON/XML**). This is why REST is known as Representational State Transfer.

REST permits different data format such as Plain text, HTML, XML, JSON, etc. But the most preferred format for transferring data is JSON.

**Soap**: simple object access protocol

SOAP can only work with XML format. As seen from SOAP messages, all data passed is in XML format.

 In SOAP, the WSDL file provides the client with the necessary information which can be used to understand what services the web service can offer.

### When we can use SOAP API?

We can use SOAP API to perform the operation on records like create, retrieve, update or delete. We can use API to manage password, perform searches etc.

### What is a RESTFul Web Services?

### Web services developed in the REST style are referred to as RESTful web services. These web services use HTTP methods to implement the concept of REST architecture(rest api send request to the server using http protocol)

**HTTP methods in API:**

Get, Put, Post, Patch, Delete.. etc.

### What is a “Resource” in REST?

REST architecture treats any content as a resource, which can be either text files, HTML pages, images, videos or dynamic business information.  
REST Server gives access to resources and modifies them, where each resource is identified by URIs/ global IDs.

# What does URI mean?

Uniform Resource Identifier (URI) is a string of characters used to identify a resource either by location (URL), or a name (URN), or both (URL and URN).

Every URL and URN is a URI, but NOT every URI is a URN and URL.

A URI’s format is <protocol>://<service-name>/<ResourceType>/<ResourceID>.

### What is URI? What is the purpose of web-based service and what is it's format?

URI stands for Uniform Resource Identifier. It is a string of characters designed for unambiguous identification of resources and extensibility by the URI scheme. The purpose of URI is to locate the resource on the server hosting of the web service.

A URIs format is **<protocol>://<service-name>/<Resource Type>/<ResourceID>**

### What is the most popular way to represent a resource in REST?

REST uses different representations to define a resource like text, JSON, and XML.  
XML and JSON are the most popular representations of resources.

### Which protocol is used by RESTful Web services?

RESTful web services use the HTTP protocol as a medium of communication between the client and the server.

**What are some key characteristics of REST?**

Key characteristics of REST are likely asked in a Web API Testing interview. So please get the answer ready in your mind with these 2 ones:

* REST is stateless, therefore the SERVER has no status (or session data), **stateless** means that server can not store any session related client data which means, everything a server needs to understand with respect to a particular resource, that is contained in a request, server is not going to store anything related to client.
* **It supports JSON and XML**
* **It is simple than SOAP**
* Web service uses POST method primarily to perform operations, while REST uses GET for accessing resources.

### What is messaging in RESTful Web services?

RESTful web services use the HTTP protocol as a communication tool between the client and the server. The technique that when the client sends a message in the form of an HTTP Request, the server sends back the HTTP reply is called Messaging. These messages comprise message data and metadata, that is, information on the message itself.

**What is http request?**

Request made by client to the server

**HTTP** (Hypertext Transfer Protocol), is the underlying format that is used to structure **request** and responses for effective communication between a client and a server.

**What is http response?**

After receiving and interpreting a request message, a server responds with an **HTTP response** message.

**What are the core components of an HTTP request?**

An HTTP request contains five key elements:

1. An action showing HTTP methods like GET, PUT, POST, DELETE.
2. Uniform Resource Identifier (URI), which is the identifier for the resource on the server.
3. HTTP Version, which indicates HTTP version, for example-HTTP v1.1.
4. Request Header, which carries metadata (as key-value pairs) for the HTTP Request message. Metadata could be a client (or browser) type, format supported by the client, format of a message body format, cache settings, and so on.
5. Request Body, which indicates the message content or resource representation.

**What are the most commonly used HTTP methods supported by REST?**

* GET is only used to request data from a specified resource.
* POST is used to send data to a server to create/update a resource.
* PUT replaces all current representations of the target resource with the request payload.
* DELETE removes the specified resource.
* OPTIONS is used to describe the communication options for the target resource.
* HEAD asks for a response identical to that of a GET request, but without the response body.

### Can GET request to be used instead of PUT to create a resource?

The PUT or POST method should be used to create a resource. GET is only used to request data from a specified resource.

**What is idempotent?**

### Get/ Put/ Delete methods are idempotent(no effect on result if we hit same request multiple times)

### Post method is non- idempotent.(Multiple requests create if we hit request multiple times)

### Multiple POST and PUT Request with same Payload?

### Idempotent and non-idempotent concept will be there

### Is there any difference between PUT and POST operations?

PUT and POST operation are quite similar, except the terms of the result generated by them.

PUT operation is idempotent(**multiple identical requests with that method is the**

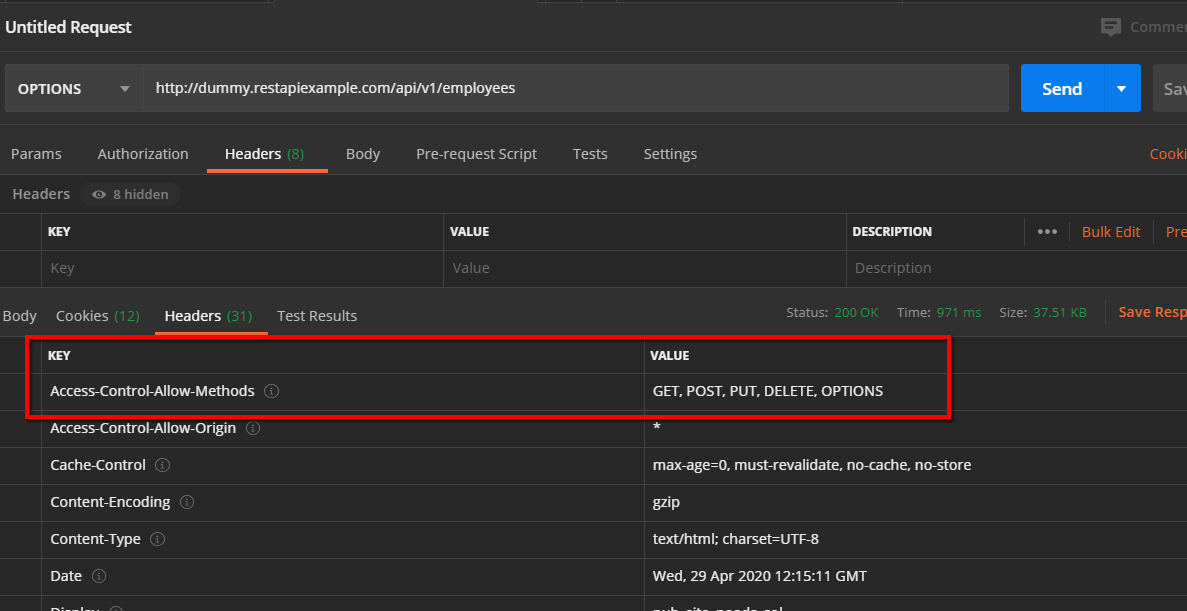
**same as the effect for a single such request,{example: PUT/DELETE}**) , so you can cache the response while the responses to POST operation are not cacheable, and if you retry the request N times, you will end up having N resources with N different URIs created on server.

In a Web API Testing interview, you should give a specific example for PUT and POST operations to make crystal clear to the interviewer. Below is an example:

Scenario: Let’s say we are designing a network application. Let’s list down few URIs and their purpose to get to know when to use POST and when to use PUT operations.  
  
GET /device-management/devices : Get all devices  
POST /device-management/devices : Create a new device  
  
GET /device-management/devices/{id} : Get the device information identified by “id”  
PUT /device-management/devices/{id} : Update the device information identified by “id”  
DELETE /device-management/devices/{id} : Delete device by “id”

### Which purpose does the OPTIONS method serve for the RESTful Web services?

The OPTIONS Method lists down all the operations of a web service supports. It creates read-only requests to the server.



API: <http://dummy.restapiexample.com/api/v1/employees>

### What is payload in Restful Web services?

The “payload” is the data you are interested in transporting.

For example, a tanker truck may carry 20 tons of oil, but the fully loaded vehicle weighs much more than that — there's the vehicle itself, the driver, fuel, the tank, etc. It costs money to move all these, but the customer only cares about (and pays for) the oil, hence, payload.

{

"status":"OK",

"data":

{

"message":"Hello, world!"

}

}

In this example, the string "Hello, world!" is the payload, the part that the recipient is interested in; the rest, while vital information, is **protocol overhead**.

**What is a payload in API?**

A payload in API is data sent with the GET method in HTTP. Typicaly a request. For example if you send a GET request to open the URL [https://www.quora.com](https://www.quora.com/), the code thats used to do so is the payload.

|  |  |
| --- | --- |
| **Web Service** | **API** |
| All web services are APIs. | All APIs are not web services. |
| It supports XML. | Responses are formatted using Web API's  MediaTypeFormatter into XML, JSON, or any other  given format. |
| You need a SOAP protocol to send or receive and data over the network. Therefore it does not have light-weight architecture. | API has a light-weight architecture. |
| It can be used by any client who understands XML. | It can be used by a client who understands JSON or  XML. |
| Web service uses three styles: REST, SOAP, and XML-RPC for communication. | API can be used for any style of communication. |
| It provides supports only for the HTTP protocol. | It provides support for the HTTP/s protocol: URL Request/Response Headers, etc. |

### What are main differences between API and Web Service?

All Web services are APIs but not all APIs are Web services.

A Web service uses only three styles of use: SOAP, REST and XML-RPC for communication whereas API may be exposed to in multiple ways(REST-**style** (Representational State Transfer), GraphQL, Falcor, the RPC **style** (Remote Procedure Call), the SOAP **style** and gRPC).

A Web service always needs a network for its operation whereas an API doesn’t need a network for its operation.

### What is the upper limit for a payload to pass in the POST method?

* The POST method allows sending far more data than the GET method, which is limited by the [URL length](https://stackoverflow.com/questions/417142/what-is-the-maximum-length-of-a-url-in-different-browsers) - about 2KB.
* The maximum [POST request body](https://stackoverflow.com/questions/14551194/how-are-parameters-sent-in-an-http-post-request) size is configured on the HTTP server and typically ranges from  
  **1MB to 2GB**
* The HTTP client (browser or other user agent) can have its own limitations. Therefore, the maximum POST body request size is min(serverMaximumSize, clientMaximumSize).

Here are the POST body sizes for some of the more popular HTTP servers:

* Ngix ([largest web server market share](https://news.netcraft.com/archives/2019/04/22/april-2019-web-server-survey.html) as of April 2019) - default [1MB](https://stackoverflow.com/questions/28476643/default-nginx-client-max-body-size/28476755#28476755), no practical maximum ([2\*\*63](https://stackoverflow.com/questions/28476643/default-nginx-client-max-body-size/28476755?noredirect=1#comment98741319_28476755))
* Apache - [maximum 2GB](https://httpd.apache.org/docs/2.4/mod/core.html#limitrequestbody), no default documented
* IIS - [default 28.6MB](https://docs.microsoft.com/en-us/iis/configuration/system.webserver/security/requestfiltering/requestlimits/#attributes) for the request length, 2048 bytes for the query string; maximum undocumented
* InfluxDB - [default ~25MB](https://docs.influxdata.com/influxdb/latest/administration/config/#max-body-size-25000000), maximum undocumented

### What is API test environment?

For API the test environment is a quite complex method where the configuration of server and database is done as per the requirement of the software application. API testing does not involve graphical user interface (GUI).

### What is API framework?

A framework or software framework is a platform for developing software applications. API framework is a foundation on which software developer can build applications for a specific platform.

**Example:** A framework can include predefined classes and functions that can be used to process input, manage hardware devices and interact with system software.

**HTTP client vs Rest Assured**

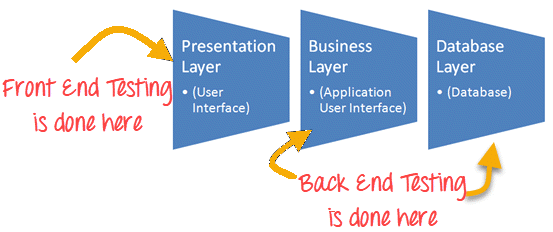
HttpClient sends request to and get response from server over HTTP protocol.

**REST Assured** is a Java library that provides a domain-specific language (DSL) for writing powerful, maintainable tests for RESTful APIs.

**Is api testing backend testing?**

Yes

**Different layers of any web application?**



**Oauth1 vs Oauth2 ?**

* 1. **OAuth 2.0 no longer requires client applications to have cryptography.(optional: oauth2 has complicated model)**

**Cryptography: Cryptography** or **cryptology** is the practice and study of techniques for [**secure communication**](https://en.wikipedia.org/wiki/Secure_communication) in the presence of third parties called [adversaries](https://en.wikipedia.org/wiki/Adversary_(cryptography)).[[2]](https://en.wikipedia.org/wiki/Cryptography#cite_note-rivest90-2) More generally, cryptography is about constructing and analyzing [protocols](https://en.wikipedia.org/wiki/Communications_protocol) that prevent third parties or the public from reading private messages.

* 1. **OAuth 2.0 signatures are much less complicated.**
  2. **OAuth 2.0 Access tokens are "short-lived".** Typically, OAuth 1.0 Access tokens could be stored for a year or more.

**What is Postman?**

Postman is a google chrome app for interacting with APIs or Postman is a API client use to develop, test, share APIs.

**What is Collection?**

Collection is a group of API requests.

**Advantages of collection?**

1. Need to define authorization(Ex: basic auth) one time only.
2. Need to define common pre-requisites, common vars, common tests one time only.

**What is swagger?**

**Swagger** is a set of open-source tools that can help you design, build, document and consume REST APIs.

**How to build request using rest assured jar?**

*given*().headers(header).when().get("/subjects").then().log().all();

**what are components of the Rest Api Automation framework?**

JAVA

Eclipse /IntelliJ IDE

GIT(Optional)

Jenkins(Optional)

TestNG

Extent reports

Rest assured

Log4J

**How to validate if your data is posted to server after we hit the post/put request through api client?**

Assert that the Response Code contains the value ‘200’

Assert that the check ID extracted from the JSON path is the expected value[JsonPath js = **new** JsonPath(res.asString());

*accessToken* = js.get("credentials.accessToken");]

**What are the different methods available in soap?**

POST ?

**Why soap is more secured than rest API?**

Although SOAP and REST both support SSL (Secure Socket Layer) for data protection, while making the request, SOAP supports Web Services Security (also known as WS- Security or WSS) for enterprise-level protection which is absent in REST Services.

**Can we automate soap services using by Java language?**

To Automate a SOAP API, you should have WSDL file which contains all the required information of the exposed APIs and is in the form of XML.

Now to automate we need to convert WSDL file to Java bean files which can be done easily using [eclipse](http://help.eclipse.org/luna/index.jsp?topic=%2Forg.eclipse.jst.ws.axis2.ui.doc.user%2Ftopics%2Ftaxis2td.html)

OR

There is tool available [SOAP UI](https://www.soapui.org/) for soap and rest API testing which you can use to automate SOAP API.

**What type of payload/body passing in API?**

Json?

**is API automation just like hit endpoint, get response?**

No, assertions is there, checking the core functionality, checking theflow, security of app can be check.

**How we can extract all the headers at the run time?**

Headers headers = res.getHeaders();

System.***out***.println("Response headers is:\n" + headers);

**How we can extract single header at the run time?**

String head= res.getHeader("Token");

**How we can extract body at the run time?**

JSONObject json = **new** JSONObject(body);

String name1 = json.get("name").toString();

What are the Client Errors?

Client errors, or HTTP status codes from 400 to 499, are the result of HTTP requests sent by a user client (i.e. a web browser or other HTTP client). Even though these types of errors are client-related, it is often useful to know which error code a user is encountering to determine if the potential issue can be fixed by server configuration.

What are the tools used in API testing?

1. **SoapUI**. **SoapUI** is a headless functional testing tool dedicated to API testing, allowing users to test REST and SOAP APIs and Web Services easily. ...
2. **Postman**. ...
3. Katalon Studio. ...
4. Tricentis Tosca. ...
5. Apigee. ...
6. JMeter. ...
7. Rest-Assured

How SOAPUI is different from Postman?

**SoapUI** is heavy tool and it is used for both rest and soap API's. **Postman** is light weight tool and it comes with chrome plugin too. This only supports REST protocols and you can only use it for rest API.

What is the role of Authorization when we are making API call?

**Authorization**: Involves checking resources that the user is **authorized** to access or modify via defined **roles** or claims. For example, the authenticated user is **authorized** for read access to a database but not allowed to modify it. The same can be applied to your **API**.

What exactly endpoint is in WebService Testing?

A **web service endpoint** is a web address (URL) at which user can gain access to access to a specific service.

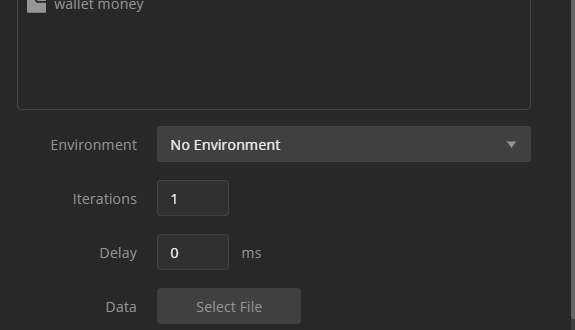
What is an API endpoint?

An **endpoint** is one end of a communication channel. When an **API interacts** with another system, the touchpoints of this communication are considered **endpoints**.

How to test multiple API call parallely?

<https://www.youtube.com/watch?v=sZKoBQ1HpKw&t=308s>

In jmeter we can increase number of threads/ In postman we can increase iterations



**How difficult it is to move from Postman to RestAssured and TestNG?**

Basic Java concepts should be needed

**Challenges we faced during API Testing??**

## ****Sequencing the API Calls??****

Why status code?

HTTP status codes let us know whether our request is a success, a failure or something in between.

**Http status code is the way in which REST developers can tackle error handling.**

In Rest API, server sends some kind of response which embeds some code and message related to that status code.

**Responses are grouped in five classes:**

1. Informational responses (100–199)
2. Successful responses (200–299)
3. Redirects (300–399)
4. Client errors (400–499)
5. Server errors (500–599)

**1XX** Informational responses: Server wants to send some information to the client.

**100 Continue:** When server send 100 it saying I am receiving your request please continue sending your request.

**101 Switching Protocols**: In this protocol Server says to the client that you are using a certain protocol but we may use a better version.

**102 Processing:** Server saying to the client, I have received your request and I am processing it, I need more time.

**103 Early hints:** It is similar to 102 but in that server not only tells the client that the request are under processing but also sends back some useful information.

**122 Request URI too long**:This status code is specific to internet explorer, If we sending a request having length more than 2032 characters then  **IE** says request URI is too long.

**2XX:** It means the action was successfully received, understood, and accepted.

**200 OK**: Request you made is a good request, it completed correctly, it did what it suppose to did.

**201 Created**: It says that client was trying to create a new resource to server and it was successful.

**202 Accepted**: It says I have accepted your request and I will process it later

[The request has been accepted for processing, but the processing has not been completed yet. ]

**203 Non-Authoritative Information :** It let the client know that there is some proxy setting between client and server and the proxy may or may not have change the response.

**204 No Content**: everything went well as expected but there is nothing to return i.e api server is not going to give you information back.

**206 Partial Content**: When Range header send from the client to request only the part of the resource.

Content-Range: bytes 200-1000/67589

**207 Multi Status** : It is used when there are multiple response codes.

**208 Already reported** : It is similar to 207 but it is used only for WebDAV

WebDAV(Web Distributed Authoring and Versioning ) is an extension of HTTP that allows system to read and write document on the web

**3XX Redirection**: It means further action must be taken in order to complete the request.

When someone goes to some website and that web page URL is redirecting into another webpage URL ( url says hey!! This page is use to be here but now it’s over there)

**300 Multiple Choices : It** indicates that the request has more than one possible responses.

**301 Moved Permanently**: Permanently replacing one URL to other.

It indicates that the resource requested has been definitively moved to the URL given by the [Location](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Location) headers.

**302 Found** : Indicates that the resource requested has been temporarily moved to the URL given by the [Location](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Location) header.

**304 Not Modified: It indicates that a file is unchanged since it was last accessed on the server**

**Browser make a request with the “If-Modified-Since” header, If the file has’nt be modifiedthen, request wont be fulfilled.**

**307 Temporary Redirect**: Temporarilyreplacing one URL to other[For ex: for summer sales we have [www.abc.com/summers](http://www.abc.com/summers) after summer we make it to [www.abc.com/winters](http://www.abc.com/winters)]

The only difference between 307 and [302](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/302) is that 307 guarantees that the method and the body of response will not be changed when the redirected request is made.

302 is then unpredictable on the Web, whereas the behavior with 307 is predictable.

**308 Permanent Redirect** : Indicates that the resource requested has been definitively moved to the URL given by the [Location](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Location) headers.

The difference between 301 and 308 is that 308 redirect is a MUST do request on the target location. If the request was a POST and and had a body, then the client must do a POST request with a body on the new location.

In the case of 301 a client may do this. In practice, most clients don’t do this and convert the POST request to a GET request.

**4XX** **Client Errors** : This category of error status codes points the finger at clients.

**400 Bad Request**: its an error from the client side, this error says ‘something went wrong’, ‘your information is bad’ and I am telling you its bad by sending 400

**401 Unauthorized**: you are trying to access something which require some type of authentication like you must be logged in or you want some API key

You didn’t pass api key/login details or paas it wrong (server send 401 and say I don’t know who you are you have to give me API key)

**403 Forbidden**: It happen when a client send a API key, but the service they’re accessing require different permission.

**404 Not Found**: you’re trying to access that part of application which don’t exist.

**408 Request Timeout** : When the request of the client takes too long, the server closes the connection and we get 408 error code.

**409 Conflict :** indicates a request conflict with current state of the server.

Conflicts are most likely to occur in response to a [PUT](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT) request. For example, you may get a 409 response when uploading a file which is older than the one already on the server resulting in a version control conflict.

**410 Gone :** indicates that access to the target resource is no longer available at the origin server and that this condition is likely to be permanent.

A 404 status code does not indicate whether the resource is temporarily or permanently missing. But if a resource is permanently removed, a [410](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/410) (Gone) should be used instead of a 404 status.

**412 Precondition Failed :** It indicates that access to the target resource has been denied.

This happens with conditional requests, when the condition defined by the [If-Unmodified-Since](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/If-Unmodified-Since) or [If-None-Match](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/If-None-Match) headers is not fulfilled.

If-None-Match: "<etag\_value>"

If-None-Match: "<etag\_value>", "<etag\_value>", …

If-None-Match: \*

**415 Unsupported Media Type :It** indicates that the server refuses to accept the request because the payload format is in an unsupported format.

**417 Expectation Failed** : indicates that the expectation given in the request's [Expect](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Expect) header could not be met.

PUT /somewhere/fun HTTP/1.1

Host: origin.example.com

Content-Type: video/h264

Content-Length: 1234567890987

Expect: 100-continue

**418 I'm a teapot :** This error is a reference to Hyper Text Coffee Pot Control Protocol defined in April Fools' jokes in 1998 and 2014.

**429 Too Many Requests :** indicates the user has sent too many requests in a given amount of time ("rate limiting").

A [Retry-After](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Retry-After) header might be included to this response indicating how long to wait before making a new request.

**451 Unavailable For Legal Reasons : It** indicates that the user requested a resource that is not available due to legal reasons, such as a web page for which a legal action has been issued.

**500 Internal server error**: Something broke on the server(for ex: your database is down for some reason )

#### ****Other Important 5xx Status Codes****

* 500 – Internal Server Error: An error that does not match other class errors.
* 501 – Not Implemented: The server doesn’t understand or cannot support the request.
* 502 – Bad Gateway: The server received an invalid message from the upstream server.

#### ****Other 5xx Status Codes****

* 504 – Gateway Timeout
* 505 – HTTP Version Not Supported
* 506 – Variant Also Negotiates
* 507 – Insufficient Storage
* 509 – Bandwidth Limit Exceeded
* 510 – Not Extended
* 511 – Network Authentication Required
* 550 – Permission Denied

SOAP vs REST

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| It is heavy weighted | It is light weighted |
| Use XML to work with | REST permits different data format such as Plain text, HTML, XML, JSON, etc |
| SOAP stands for Simple Object Access Protocol | REST stands for Representational State Transfer |
| SOAP is a protocol | REST is an Architectural style |
| It includes a WSDL file which has the required information on what the web service | REST is stateless: **stateless** means that server can not store any session related client data which means, everything a server needs to understand with respect to a particular resource, that is contained in a request, server is not going to store anything related to client. |

**URI vs URL vs URN**

Every URL and URN is URI

**URI**: uniform resource identifier

It is a string of characters use to identify a resource on the internet either by location or name or by both.

Name : ashish

Address: uttam nagar

**URL:** Uniform resource locator we can identify resource using location only

Address: uttam nagar

url contains:

<protocol>://<domain>

Below are optional path url may contains:

**Path/port/other components**

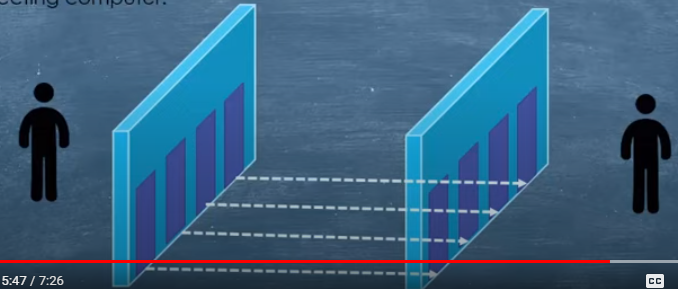
**PATH**

**EX:** http://www.myweb.com**/products/product.php**

**Path: /products/product.php**

**PORT:** <http://www.myweb.com:80>

Port: choose the door to connect



<http://www.myweb.com:80>

**Others: query string-** $id=1&name=trimmer

<http://www.myweb.com/product$id=1&name=trimmer>

**URN:** Uniform resource name we can identify resource using name only

Name : ashish

Best example of URN is ISBN(International Standard Book Number)



**How to debug your app/web?**

Charles

**Webservices Vs API:**

**Webservices:**

* All APIs are webservices but not all webservices are API.
* Responsible for interaction between two machines **over a network**.
* A webservies is a api wrapped with HTTP(all api I am using now days are restful webservices)
* Webservices design using three styles ex: Soap, rest, XML-RPC

**API**:

* Responsible for interaction between two applications
* Like when we work on ms office no network connection is required so, here we use APIs not webservices

Our libraries(JAR files) are comes under API

* APIs may use any style for communication