REST - REpresentational State Transfer

Tutorialpoint

REST Server simply provides access to resources and REST client accesses and modifies the resources.

HTTP methods –

POST – Read only access to a resource

GET – Creates a new resource

DELETE – Deletes a resource

PUT – modifies or creates a new resource

Web services based on REST architecture are known as RESTful services.

REST architecture treats every content as resource.

RESTful Web Services make use of HTTP protocols as a medium of communication between client and server. A client sends a message in form of a HTTP Request and the server responds in the form of an HTTP Response.

HTTP Request –

5 parts

Verb – GET, PUT, DELETE, POST – Indicates the HTTP method

URI – Uniform Resource Identifier – Indicates resource on the server

HTTP Version

Request Header

Request Body

HTTP Response –

4 parts

Response code

HTTP Version

Response header

Response Body

URI format :

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

A RESTful webservice should not keep client’s state in Server. This is called as Statelessness.

Caching – Storing server response for a certain period so that client does not have to make request to the server repeatedly.

Server provides information to the client about caching.

JAX RS – Java API for RESTful web services.

Some of the annotation are:

@GET, @PATH, @PUT, @POST, @HEAD, @Produces, @Consumes, @PathParam, @Context, @DefaultValue, @HeaderParam etc

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

SOAP - Simple Object Access Protocol

Is an XML based messaging protocol to exchange information among computers.

SOAP message has following elements –

Envelope – Mandatory element – defines start and end of a message.

Header – Optional – attributes of the message used in processing of the message

Body – Mandatory – contains the XML data that is being sent.

Fault -- optional – provides information about the error that occurs while processing the message.

<?xml version = "1.0"?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"

SOAP-ENV:encodingStyle = "http://www.w3.org/2001/12/soap-encoding">

<SOAP-ENV:Header>

...

...

</SOAP-ENV:Header>

<SOAP-ENV:Body>

...

...

<SOAP-ENV:Fault>

...

...

</SOAP-ENV:Fault>

...

</SOAP-ENV:Body>

</SOAP\_ENV:Envelope>

SOAP with HTTP POST method

POST /OrderEntry HTTP/1.1

Host: www.tutorialspoint.com

Content-Type: application/soap; charset="utf-8"

Content-Length: nnnn

<?xml version = "1.0"?>

<SOAP-ENV:Envelope

xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"

SOAP-ENV:encodingStyle = " http://www.w3.org/2001/12/soap-encoding">

...

Message information goes here

...

</SOAP-ENV:Envelope>

SOAP header can have 2 attributes-

Actor – message path as SOAP servicing nodes. These nodes can perform some tasks before the message is sent for processing

MustUnderstand – indicates if header is optional or not. If it is true the recipient must understand the header.

Fault – 200 -299 codes are success codes

500-599 codes are linked to fault codes

SOAP data types –

scalar (contains one value )  
 compound ( multiple value like array and struct)

Microservices

Microservice architecture – In this architecture smaller services are built and delivered and they are loosely coupled with other services and can be developed independently with no major side effects.

The smaller size of microservices enables developers to be more productive.

It’s faster and easier for developers to understand and test each microservice. They are stronger such that developers can correctly handle failure of any dependent service, and they reduce the impact of correlated failures.

Backend

Backend development languages handle the ‘behind-the-scenes’ functionality of web applications. It’s code that connects the web to a database, manages user connections, and powers the web application itself. Backend development works in tandem with the front end to deliver the final product to the end user.

Some of the Backend languages are:

JAVA, JavaScript, PHP, C#, VB, .Net, Ruby, Python, SQL

Frontend

Front-end web development, also known as client-side development is the practice of producing HTML, CSS and JavaScript for a website or Web Application so that a user can see and interact with them directly.

A front-end developer architects and develops websites and web applications using web technologies (i.e., HTML, CSS, and JavaScript), which typically runs on the Open Web Platform or acts as compilation input for non-web platform environments (i.e., React Native).

A person enters into the field of front-end development by learning to build a website or web application which relies on HTML, CSS, and JavaScript and commonly runs in a web browser but can also run in a headless browser, WebView, or as compilation input for a native runtime environment.

Headless browsers are a web browser **without** a graphical user interface that can be controlled from a command line interface programmatically for the purpose of web page automation (e.g., functional testing, scraping, unit testing, etc.)

Client Server Architecture

In client server architecture the server can be defined as a provider of services, and the client can be a requester to services.

Two Tier-Architecture

In the two-tier architecture, there exists only two parties; one is the client, and the other is the server.

### Three-Tier / Multi tier Architecture

In this architecture, there is a middle layer within the client and the server. It clarifies the difficulty of waiting time by clients. It maintains a queue of requests by different clients, and the client can do their work; meanwhile, the server is busy processing other client’s request. The middle layer forwards the request, and the server, in turn, replies to the client.