REST Architectural Style

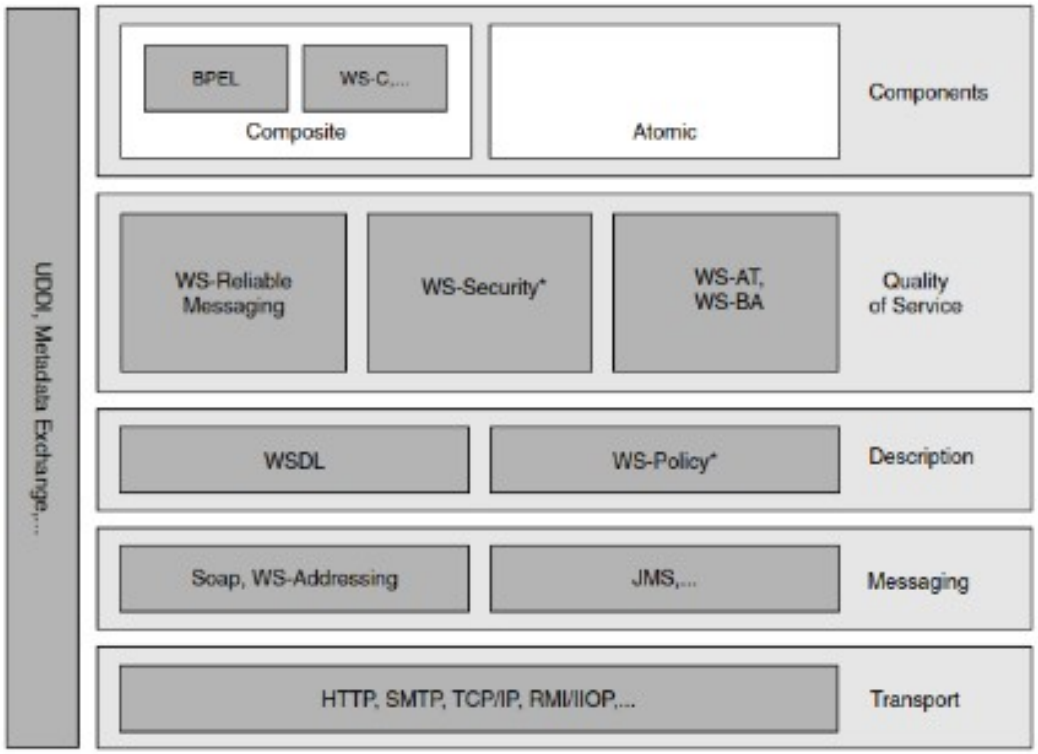
Simple Object Access Protocol (SOAP)

- **SOAP is a method of transferring messages, or small amounts of information, over the Internet. SOAP messages are formatted in XML and are typically sent using HTTP (hypertext transfer protocol).**

- WSDL defines contract between client and service and is static by its nature.
- SOAP builds an XML based protocol on top of HTTP or sometimes TCP/IP.
- SOAP describes functions, and types of data.
- SOAP is a successor of XML-RPC and is very similar, but describes a standard way to communicate.
- Several programming languages have native support for SOAP, you typically feed it a web service URL and you can call its web service functions without the need of specific code.
- Binary data that is sent must be encoded first into a format such as base64 encoded.
- Has several protocols and technologies relating to it: WSDL, XSDs, SOAP, WS-Addressing.

Simple Object Access Protocol (SOAP)

- **SOA consists of many layers including protocols and standards for security and reliability therefore, is complicated to design**



- **SOAP is useful for domains that require various qualities of service**

Representational State Transfer (REST)

- Architectural style for providing standards between computer systems on the web making them easier to communicate.
- Introduced and Collated by Dr. Roy Fielding's thesis :
  Architectural Styles and Design of Network Based Software Architectures

- Resource Based
  - Things Vs. Actions
  - Nouns vs. Verbs
  - Identified by URIs
  - Separate from their representations
- REST server provides access to resource and REST clients accesses and modifies resources
- Each resource is identified by URIs and uses various representations to represent resources
- HTTP methods are used in REST architectures
- REST is preferable for domains which are query intense and require exchange of large chunks of data

Representational State Transfer (REST)

- REST is best used with Resource Oriented Architecture (ROA). In this mode of thinking everything is a resource, and you would operate on these resources.

- Simple way of sending and receiving data between client and server and it doesn't have very many standards defined - send and receive data as JSON, XML or even plain text.

- REST is very lightweight - don't need all of this complexity that SOAP created.

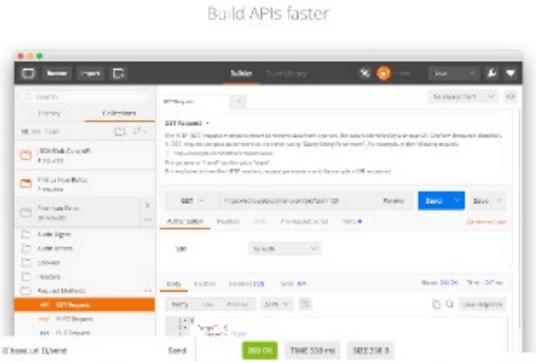
- REST is a very simple in that it uses HTTP GET, POST and PUT methods to update resources on the server instead of a big XML format describing everything.

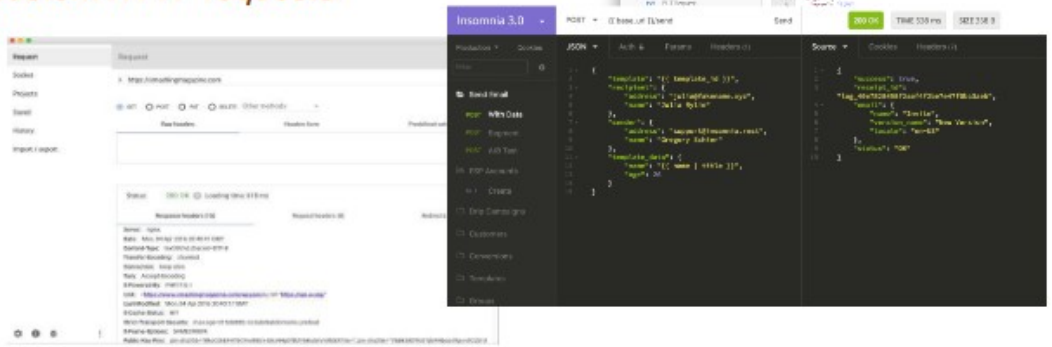
- As long as the programming language has an HTTP library (most do) you can consume a REST HTTP protocol very easily.
- Binary data or binary resources can simply be delivered upon their request.

- In case of REST contract between client and service is somewhat complicated and is defined by HTTP, URI, Media Formats and Application Specific Coordination Protocol. It's highly dynamic unlike WSDL.

REST Web Services

REST API Client *is the web developers helper program to create and test custom HTTP requests.*





Chrome

Postman: https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop?hl=en

Advanced REST client: https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjeloo

DHC: https://chrome.google.com/webstore/detail/dhc-rest-client/aejoelaoggembcahagimdiliamlcdmfm?hl=en

Firefox

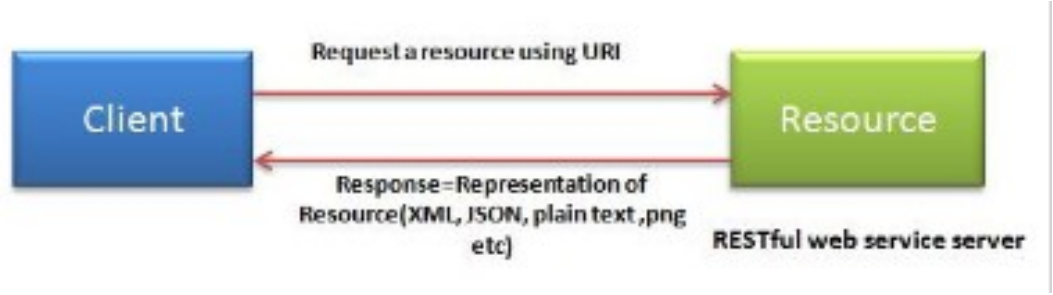
Firefox add-on: https://addons.mozilla.org/en-US/firefox/addon/restclient/

Insomnia *is a cross-platform application for organizing, running, and debugging HTTP requests (https://insomnia.rest/).*

RESTful Web Services

Representational State Transfer (REST) is a stateless client-server architecture in which the web services are viewed as resources and can be identified by their URIs.

- Web service clients that want to use these resources access via globally defined set of remote methods that describe the action to be performed on the resource.

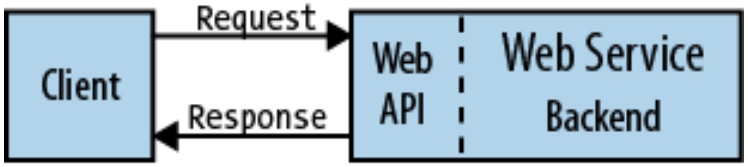


Clients and servers exchange representations of resources by using a standardized interface and protocol.

- RESTFul web services are based on HTTP methods and the concept of REST
- A concrete implementation of a REST Web service follows four basic design principles:
- Use HTTP methods explicitly.
- Be stateless.
- Expose directory structure-like URIs.
- Transfer XML, JavaScript Object Notation (JSON), or both.

REST APIs

- Web services are purpose-built web servers that support the needs of a site or any other application.
- Client programs use application programming interfaces (APIs) to communicate with web services.
- API exposes a set of data and functions to facilitate interactions between computer programs and allow them to exchange information
- Web API is the face of a web service, directly listening and responding to client requests.



- A Web API conforming to the REST architectural style is a REST API.
- Having a REST API makes a web service “RESTful.”
- A REST API consists of an assembly of interlinked resources. This set of resources is known as the REST API’s resource model.

Guiding Principles for API service to be RESTful

- Use of uniform interface (UI)
- Client-server based
- Stateless operations
- RESTful resource caching
- Layered system
- Code on demand

Representations

- **How resources get modified**
- **Part of resource state**
  - Transferred between client and server
- **Typically JSON or XML**
- **Example :**
  - Resource : Person
  - Service : Contact Information (*GET* - HTTP Verb)
  - Representation :
    - Name , address, phone number
    - JSON, XML

Uniform Interface

- **Defines interface between client and Server**
- **Simplifies and decouples the architecture**
- **Fundamentals to RESTful design**
  - HTTP verbs
    - GET - retrieve a specific resource (by id) or collection of resources
    - POST - create a new resource
    - PUT - update specific resource (id)
    - DELETE – remove a specific resource by id
  - URIs (resource name)
  - HTTP Response (status , body)

Stateless

- **Server contains no client state**
- **Each request contains enough context to process the message**
  - Self descriptive messages
- **Any session state is held on the client**

Client - Server

- Assume a disconnected system
- Separation of concerns
- Uniform interface is the link between the two

Cacheable

- **Server responses (responses) are cacheable**
  - Implicitly
  - Explicitly
  - Negotiated

Layered System

- Client can’t assume direct connection to server
- Software or hardware intermediaries between client and server
- Improves scalability

Code on Demand

- Server can temporarily extend client
- Transfer logic to client
- Client executes logic
- For example : Java Applets, Java Scripts
- The only optional constraint

Summary

- **Violating any constraint other than Code on Demand means service is not strictly RESTful**
  - Example : three legged OAUTH2
- **Compliance with REST constraints allows :**
  - Scalability
  - Simplicity
  - Modifiability
  - Visibility
  - Portability
  - Reliability