REST based Web Services

- Overall **Web Services architecture** consists of many layers including **protocols and standards for security and reliability**.
- Therefore it becomes **complicated and time/resource consuming** for developers of simple web services.
- **Web Services** can be implemented in Web environments too, on top of basic Web technologies such as HTTP, Simple Mail Transfer Protocol (SMTP), and so on.

REpresentational State Transfer (REST) has gained widespread acceptance across the Web as a simpler alternative to SOAP and WSDL based Web services.

REST defines a set of architectural principles by which you can design Web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages.

Simply put, it is the architecture of the Web.

REST has emerged in the last several years alone as a predominant Web service design model.

REST has had such a large impact on the Web that it has mostly displaced SOAP- and WSDL-based interface design because it’s a considerably simpler style to use.

REST is preferable in problem domains that are query intense or that require exchange of large grain chunks of data.

Basically, you would want to use RESTful web services for integration over the web.

SOAP and WSDL based Web services (“Big” Web services) are preferable in areas that require asynchrony and various qualities of services.

Finally, SOAP-based custom interfaces enable straightforward creation of new services based on choreography.

Therefore, you will use big web services in enterprise application integration scenarios that have advanced quality of service (QoS) requirements.

What is REST?

“Representational State Transfer is intended to evoke an image of how a well-designed web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use.

REST- An Architectural Style of Networked System

Underlying Architectural model of the World Wide Web.

Guiding framework for Web protocol standards.

REST based web services

Online shopping

Search services

Dictionary services.

Characteristics of a REST based network

Client-Server: a pull-based interaction style (Client request data from servers as and when needed).

Stateless: each request from client to server must contain all the information necessary to understand the request, and cannot take advantage of any stored context on the server.

Cache: to improve network efficiency, responses must be capable of being labelled as caheable or non-cacheable.

Uniform interface: all resources are accessed with a generic interface (e.g., HTTP GET, POST, PUT, DELETE).

Named resources: the system is comprised of resources which are named using a URL.

Interconnected resource representation: the representations of the resources are interconnected using URLs, thereby enabling a client to progress from one state to another.

The characteristics of the restful web services are as below:

The restful web services have a pull-based client-server interaction manner.

Every request from the client to the server must possess the required data to understand the request and it cannot take the profit of any saved content on the server.

To advance or develop the network efficiency outputs it must be capable of being labelled as cacheable or non-cacheable.

Principles of REST web service design

- Identify all the conceptual entities that we wish to expose as services. (Examples we saw include resources such as : parts list, detailed part data, purchase order)
- Create a URL to each resource.
- Categorize our resources according to whether clients can just receive a representation of the resource (using an HTTP GET), or whether clients can modify (add to) the resource using HTTP POST, PUT and/or DELETE).

- All resources accessible via HTTP GET should be side-effect free. That is, the resource should just return a representation of the resource. Invoking the resource should not result in modifying the resource.
- Put hyperlinks within resource representations to enable clients to drill down for more information, and/or to obtain related information.
- Design to reveal data gradually. Don’t reveal everything in a single response document. Provide hyperlinks to obtain more details.
- Specify the format of response data using a schema (DTD, W3C Schema, RelaxNG or Schematron). For those services that require a POST or PUT to it, also provide a schema to specify the format of the response.
- Describe how our services are to be invoked using either a WDL document or simply an HTML document.

Principles of Restful Web Services:

The principles of the restful web services are as follows:

- REST network is a **key for designing the web services.**
- It requires or needs to **create a URL for every device**.
- The resources have to be **arranged** in a format that helps the client.
- The resources accessible through HTTP should be **free of side effect.**
- The **hyperlinks** should be kept within the representation of resources.
- The design should be in a manner that **reveals the information gradually**.

Advantages of Restful Web Services:

The advantages of restful web services are as follows:

- The restful web services have the scalable equipment interactions.
- It possesses the general interfaces.
- It can independently deploy the connections.
- It decreases the interaction latency.
- It strengthens or improves the security.
- It has the feature of safe encapsulation of legacy systems.
- It sustains the proxies and gateways as information transformation and caching equipment.
- Restful web services scale to a huge number of clients.
- It enables the alienate of information in streams that possess infinite size.

Disadvantages of Restful Web Services:

The disadvantages of restful web services are as follows:

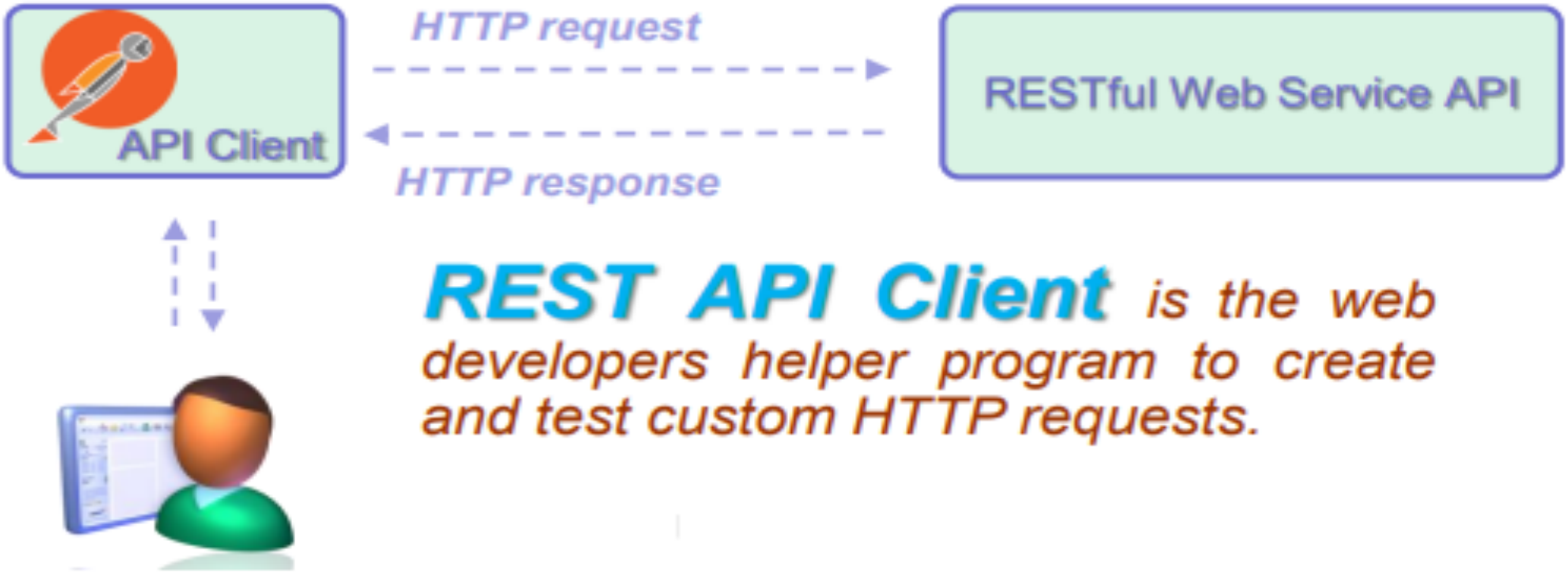
- It destroys few advantages of other architectures.
- Restful web services have a state of interaction with an FTP site.
- It contains a single interface for everything.
- It reduces the performances of the new by enhancing the repetitive information.

Why are Restful Web Services so popular?

Restful Web Services have become popular due to the below-mentioned reasons:

- Its heterogeneous language environment has made it very popular.
- It assists web applications in different programming languages for better communication.
- The web applications reside in a variety of environments with the assistance of Restful services.

REST Web Services



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

RESTful Web Services

•In the web services terms, 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.

- In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol.
- REST isn't protocol specific, but when people talk about REST they usually mean REST over HTTP.
- The response from server is considered as the representation of the resources (that can be generated from one resource or more number of resources).

- RESTFul web services are based on HTTP methods and the concept of REST. A RESTFul web service typically defines the base URI for the services, the supported MIME-types (XML, text, JSON, user-defined, ...) and the set of operations (POST, GET, PUT, DELETE) which are supported. A concrete implementation of a REST Web service follows four basic design principles:

q Use HTTP methods explicitly.

q Be stateless.

q Expose directory structure-like URIs.

q Transfer XML, JavaScript Object Notation (JSON), or both.

JAX-RS - Java API for RESTful Web Services, is a set of APIs to develop REST service. JAX-RS is part of the Java EE, and make developers to develop REST web application easily. There are two main implementation of JAX-RS API: Jersey and RESTEasy Jersey is the open source, production quality,

JAX-RS (JSR 311) Reference Implementation for building RESTful Web services. But, it is also more than the Reference Implementation. Jersey provides an API so that developers may extend Jersey to suit their needs.

Differences between SOAP and REST

No	SOAP	REST
1.	SOAP is a protocol	REST is an architectural style.
2.	SOAP stands for Simple Object Access Protocol.	REST stands for REpresentational State Transfer.
3.	SOAP can't use REST because it is a protocol.	REST can use SOAP web services because it is a concept and can use any protocol like HTTP, SOAP
4.	SOAP uses services interfaces to expose the business logic	REST uses URI to expose business logic.
5.	JAX-WS is the java API for SOAP web services.	JAX-RS is the java API for RESTful web services.

6.	SOAP defines standards to be strictly followed.	REST does not define too much standards like SOAP.
7.	SOAP requires more bandwidth and resources than REST.	REST requires less bandwidth and resources than SOAP.
8.	SOAP defines its own security.	RESTful web services inherits security measures from the underlying transport.
9.	SOAP permits XML data format only	REST permits different data format such as Plain text, HTML, XML, JSON etc.
10.	SOAP is less preferred than REST	REST more preferred than SOAP

SOAP vs REST based Web Services

SOAP Web Services

Advantages:

WS Security: SOAP defines its own security known as WS Security which adds some enterprise security features. Supports identity through intermediaries, not just point to point (SSL). It also provides a standard implementation of data integrity and data privacy.

Atomic Transaction: SOAP supports ACID (Atomicity, Consistency, Isolation, Durability) compliant transactions. Internet apps generally don't need this level of transactional reliability, enterprise apps sometimes do.

Reliable Messaging: SOAP has successful/retry logic built in and provides end-to-end reliability even through SOAP intermediaries.

Disadvantages:

Slow: SOAP uses XML format that must be parsed to be read. It defines many standards that must be followed while developing the SOAP applications.

So it is slow and **consumes more bandwidth and resource**.

WSDL dependent: SOAP uses WSDL and doesn't have any other mechanism to discover the service.

RESTful Web Services

Advantages:

Fast: **RESTful Web Services are fast because there is no strict specification like SOAP. It consumes less bandwidth and resource.**

Can use SOAP: **RESTful web services can use SOAP web services as the implementation.**

Permits different data format: **RESTful web service permits different data format such as Plain Text, HTML, XML and JSON.**

Disadvantages:

ACID Transactions: REST supports transactions, but it isn't as comprehensive and isn't ACID compliant.

REST is limited by HTTP itself which can't provide two-phase commit across distributed transactional resources, but SOAP can.

WS Security: REST supports just a standard security technology for establishing an encrypted link between a server and a client SSL (Secure Sockets Layer).

Reliable Messaging: REST doesn't have a standard messaging system and expects clients to deal with communication failures by retrying.

even though REST is more popular and preferable for Web apps and Web based services, SOAP is still useful and important for some enterprise solutions and highly reliable applications (e.g. bank clients, etc.)

REST Web Services are:

Modern

Light weight

Use a lot of concepts behind HTTP

When to choose REST?

•REST should be chosen when you have to develop a highly secure and complex API, which supports different protocols.

•Although SOAP may be a good choice, REST may be better when you have to develop lightweight APIs with great performance and support for CURD operations.