Rest VS Soap

|  |  |  |
| --- | --- | --- |
| **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 resource than REST. | REST **requires less bandwidth** and resource 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. |

**13) Mention what is JAX-WS and JAX-RS?**

Both JAX-WS and JAX-RS are libraries (APIs) for doing communication in various ways in Java.  JAX-WS is a library that can be used to do SOAP communication in JAVA, and JAX-RS lets you do the REST communication in JAVA.

**14) List out the tools or API for developing or testing web api?**

Testing tools for web services for REST APIs includes

* Spring REST web service using MVC
* Jersey API
* CFX
* Axis
* Restlet,

Difference between RPC and Document web services

There are many differences between RPC and Document web services. The important differences between RPC and Document are given below:

RPC Style

1) RPC style web services use method name and parameters to generate XML structure.

2) The generated WSDL is **difficult to be validated** against schema.

3) In RPC style, SOAP **message is sent as many elements**.

4) RPC style message is **tightly coupled**.

5) In RPC style, SOAP message **keeps the operation name**.

6) In RPC style, parameters are sent as **discrete values**.

Let's see the RPC style generated WSDL file.

**WSDL file:**

In WSDL file, it doesn't specify the types details.

1. <types/>

For message part, it defines name and type attributes.

1. <message name="getHelloWorldAsString">
2. <part name="arg0" type="xsd:string"/>
3. </message>
4. <message name="getHelloWorldAsStringResponse">
5. <part name="return" type="xsd:string"/>
6. </message>

For soap:body, it defines use and namespace attributes.

1. <binding name="HelloWorldImplPortBinding" type="tns:HelloWorld">
2. <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
3. <operation name="getHelloWorldAsString">
4. <soap:operation soapAction=""/>
5. <input>
6. <soap:body use="literal" namespace="http://javatpoint.com/"/>
7. </input>
8. <output>
9. <soap:body use="literal" namespace="http://javatpoint.com/"/>
10. </output>
11. </operation>
12. </binding>

Document Style

1) Document style web services **can be validated against predefined schema**.

2) In document style, SOAP message is **sent as a single document**.

3) Document style message is **loosely coupled**.

4) In Document style, SOAP message **loses the operation name**.

5) In Document style, parameters are sent in **XML format**.

Let's see the Document style generated WSDL file.

**WSDL file:**

In WSDL file, it specifies types details having namespace and schemaLocation.

1. <types>
2. <xsd:schema>
3. <xsd:**import** namespace="http://javatpoint.com/" schemaLocation="http://localhost:7779/ws/hello?xsd=1"/>
4. </xsd:schema>
5. </types>

For message part, it defines name and element attributes.

1. <message name="getHelloWorldAsString">
2. <part name="parameters" element="tns:getHelloWorldAsString"/>
3. </message>
4. <message name="getHelloWorldAsStringResponse">
5. <part name="parameters" element="tns:getHelloWorldAsStringResponse"/>
6. </message>

For soap:body, it defines use attribute only not namespace.

1. <binding name="HelloWorldImplPortBinding" type="tns:HelloWorld">
2. <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
3. <operation name="getHelloWorldAsString">
4. <soap:operation soapAction=""/>
5. <input>
6. <soap:body use="literal"/>
7. </input>
8. <output>
9. <soap:body use="literal"/>
10. </output>
11. </operation>
12. </binding>

Q=difference between ajax and rest call?

|  |  |
| --- | --- |
| **AJAX** | **REST** |
| * In Ajax, the request are sent to the server by using XMLHttpRequest objects. The response is used by the JavaScript code to dynamically alter the current page * Ajax is a set of technology; it is a technique of dynamically updating parts of UI without having to reload the page * Ajax eliminates the interaction between the customer and server asynchronously * REST requires the interaction between the customer and server | * REST have a URL structure and a request/response   pattern the revolve around the use of resources   * REST is a type of software architecture and   a method for users to request data or information  from servers   * REST requires the interaction between the   customer and server |

**4) Mention what are the HTTP methods supported by REST?**

HTTP methods supported by REST are:

* **GET:**It requests a resource at the request URL. It should not contain a request body as it will be discarded. May be it can be cached locally or on the server.
* **POST:**It submits information to the service for processing; it should typically return the modified or new resource
* **PUT:**At the request URL it update the resource
* **DELETE:**At the request URL it removes the resource
* **OPTIONS:**It indicates which techniques are supported
* **HEAD:**About the request URL it returns meta information

What are webservices?

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer.

What are RESTful webservices?

Web services based on REST Architecture are known as RESTful web services. These web services use HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI, Uniform Resource Identifier a service, provides resource representation such as JSON and set of HTTP Methods.

What is a Resource in REST?

REST architecture treats every content as a resource. These resources can be text files, html pages, images, videos or dynamic business data.

How to represent a resource in REST?

REST uses various representations to represent a resource where text, JSON, XML. XML and JSON are the most popular representations of resources.

Which protocol is used by RESTful webservices?

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

What is messaging in RESTful webservices?

A client sends a message in form of a HTTP Request and server responds in form of a HTTP Response. This technique is termed as Messaging. These messages contain message data and metadata

What is URI?

URI stands for Uniform Resource Identifier. Each resource in REST architecture is identified by its URI.

What is statelessness in RESTful Webservices?

As per REST architecture, a RESTful web service should not keep a client state on server. This restriction is called statelessness. It is responsibility of the client to pass its context to server and then server can store this context to process client's further request. For example, session maintained by server is identified by session identifier passed by the client.

What are the advantages of statelessness in RESTful Webservices?

Following are the benefits of statelessness in RESTful web services −

* Web services can treat each method request independently.
* Web services need not to maintain client's previous interactions. It simplifies application design.
* As HTTP is itself a statelessness protocol, RESTful Web services work seamlessly with HTTP protocol.

What are the disadvantages of statelessness in RESTful Webservices?

Following is the disadvantage of statelessness in RESTful web services −

Web services need to get extra information in each request and then interpret to get the client's state in case client interactions are to be taken care of.

What do you mean by idempotent operation?

Idempotent operations means their result will always same no matter how many times these operations are invoked.

Which type of Webservices methods are to be idempotent?

PUT and DELETE operations are idempotent.

What is the difference between PUT and POST operations?

PUT and POST operation are nearly same with the difference lying only in the result where PUT operation is idempotent and POST operation can cause different result.

What should be the purpose of OPTIONS method of RESTful web services?

It should list down the supported operations in a web service and should be read only.

What should be the purpose of HEAD method of RESTful web services?

It should return only HTTP Header, no Body and should be read only.

**What is the purpose of HTTP Status Code?**

HTTP Status code are standard codes and refers to predefined status of task done at server. For example, HTTP Status 404 states that requested resource is not present on server.

**What HTTP Status Code 200 states?**

It means, OK, shows success.

**What HTTP Status Code 201 states?**

It means, CREATED, when a resource is successful created using POST or PUT request. Return link to newly created resource using location header.

**What HTTP Status Code 204 states?**

It means, NO CONTENT, when response body is empty for example, a DELETE request.

**What HTTP Status Code 304 states?**

It means, NOT MODIFIED, used to reduce network bandwidth usage in case of conditional GET requests. Response body should be empty. Headers should have date, location etc.

**What HTTP Status Code 400 states?**

It means, BAD REQUEST, states that invalid input is provided e.g. validation error, missing data.

**What HTTP Status Code 401 states?**

It means, FORBIDDEN, states that user is not having access to method being used for example, delete access without admin rights.

**What HTTP Status Code 404 states?**

It means, NOT FOUND, states that method is not available.

**What HTTP Status Code 409 states?**

It means, CONFLICT, states conflict situation while executing the method for example, adding duplicate entry.

**What HTTP Status Code 500 states?**

It means, INTERNAL SERVER ERROR, states that server has thrown some exception while executing the method.

**What are the best practices to be followed while designing a secure RESTful web service?**

As RESTful web services work with HTTP URLs Paths so it is very important to safeguard a RESTful web service in the same manner as a website is be secured. Following are the best practices to be followed while designing a RESTful web service −

* **Validation** − Validate all inputs on the server. Protect your server against SQL or NoSQL injection attacks.
* **Session based authentication** − Use session based authentication to authenticate a user whenever a request is made to a Web Service method.
* **No sensitive data in URL** − Never use username, password or session token in URL , these values should be passed to Web Service via POST method.
* **Restriction on Method execution** − Allow restricted use of methods like GET, POST, DELETE. GET method should not be able to delete data.
* **Validate Malformed XML/JSON** − Check for well formed input passed to a web service method.
* **Throw generic Error Messages** − A web service method should use HTTP error messages like 403 to show access forbidden etc.

**What is caching?**

Caching refers to storing server response in client itself so that a client needs not to make server request for same resource again and again. A server response should have information about how a caching is to be done so that a client caches response for a period of time or never caches the server response.

**Which header of HTTP response, provides the date and time of the resource when it was created?**

Date header provides the date and time of the resource when it was created.

**Which header of HTTP response, provides the date and time of the resource when it was last modified?**

Last Modified header provides the date and time of the resource when it was last modified.

**Which header of HTTP response provides control over caching?**

Cache-Control is the primary header to control caching.

**Which header of HTTP response sets expiration date and time of caching?**

Expires header sets expiration date and time of caching.

**Which directive of Cache Control Header of HTTP response indicates that resource is cachable by any component?**

Public directive indicates that resource is cachable by any component.

**Which directive of Cache Control Header of HTTP response indicates that resource is cachable by only client and server, no intermediary can cache the resource?**

Private directive indicates that resource is cachable by only client and server, no intermediary can cache the resource.

**Which directive of Cache Control Header of HTTP response indicates that resource is not cachable?**

no-cache/no-store directive indicates that resource is not cachable.

**Which directive of Cache Control Header of HTTP response can set the time limit of caching?**

max-age directive indicates that the caching is valid up to max-age in seconds. After this, client has to make another request.

**Which directive of Cache Control Header of HTTP response provides indication to server to revalidate resource if max-age has passed?**

must-revalidate directive provides indication to server to revalidate resource if max-age has passed.

**Q=What are the best practices for caching?**

Always keep static contents like images, css, JavaScript cacheable, with expiration date of 2 to 3 days. Never keep expiry date too high.

## When to Use REST Web Services

RESTful web service is desirable to use over the web. It provides loose coupling, scalability and simplicity. In web application RESTful web service is suitable because many clients can consume it with other web based services. Find the different scenarios when to use REST web service.   
  
1. Use REST when basic integration is required. REST web service works over HTTP protocol and use existing W3C. REST is lightweight and needs minimal configuration to setup. This is because REST web service is very low expensive and can easily be adopted.   
  
2. REST web service is stateless. Suppose we need to get some data from web service and do some manipulation and again send it back to web service to get response. So if there is the case that connection can break. In this case REST web service is most suitable because REST web service is stateless and it connects for every request like normal webpage over HTTP.   
  
3. Another scenario is when the response is not dynamically generated. So in this case we can cache the response and performance can be increased highly. Caching can be achieved easily in RESTful web services.   
  
4. When the producer and consumer agrees mutually on the content being passed and there is not any tight format to exchange the data, then RESTful web service will be suitable.   
  
5. In case of mobile and PDAs, there is limitation of bandwidth. In that case RESTful web service is best option. RESTful web service is lightweight because the data is not filled in herders and also there is no overload of SOAP elements.   
  
6. RESTful web services can easily be used with AJAX. In that case a developer has not to learn new things a lot to work with it and he can easily start and develop fast.

## When to Use SOAP Web Services

SOAP based web services is desirable in case of enterprise application where advance services need to integrate with it. SOAP retains the state of transaction. Find some scenarios where we should use SOAP based web service.   
  
1. SOAP based web services interacts over XML data which is well defined and standard message format that uses web service description language (WSDL). Use SOAP when data can be consumed in desirable languages.   
  
2. In the scenario where multiple calls are requested to complete a task, SOAP will be desirable because data transfer in SOAP is faster. Once request is failed, SOAP will automatically try to complete the transaction.   
  
3. For security point of view, SOAP based web service is robust. In application where security has high measurement, then use SOAP based web service.   
  
4. SOAP implements standard exchange format and is more ideal for enterprise application. In case of complex contract between producer and consumer to transfer message, use SOAP based web service.   
  
5. SOAP based web service is supported by different tools. So in the development environment where different tools are being used, then SOAP will be better choice.   
  
6. SOAP supports SMTP, JSM, TCP etc transport protocol. So in scenario where the above protocol is being used, SOAP web service will be the choice over REST.