

# RESTful Web Services

**Abstract**—Representational State Transfer (REST) is a style of software architecture for distributed hypermedia systems that was introduced by Roy Thomas Fielding on his doctoral dissertation [3]. RESTful is a web service implementation that is using principles of REST architecture. Instead of using SOAP web services, it is getting more popular and commonly used by community. It didn't attract this much attention when it was first introduced but today major frameworks support REST and still being developed. REST defines the architecture of the web and also explains the success and scalability of HTTP protocol.

## I. INTRODUCTION

RESTful web services are web service implemented that is using HTTP and the principles of REST. RESTful is not a standard unlike SOAP-based web services but it does use standards such as http, url, xml,html and mime types[6].W3C [7] does not accept REST as a specification and many “Big Vendors” still does not release any REST developer's toolkit. It can be said that RESTful style of web services was a response to the more heavyweight SOAP-based web services.

Scalability of component interactions, generality of interfaces, independent deployment, reduce interaction latency, enforce security are the goals of the RESTful Web Services. In order to achieve these goals, RESTful approach is composed of five concepts (resource, representation, uniform identifier, unified interface, and execution scope).

### A. Research Contributions

In this paper, we classify existing standards proposals into two categories: REST-oriented and SOAP-oriented standards. We take glance at two different design of web service and focus on the RESTful Web services that use principles of REST architecture. Restful web services are gained greatly acceptance nowadays. Several major web services for instance Twitter, Yahoo, Amazaon, Ebay use RESTful as

services. Strengths and weaknesses and usage of RESTful web services are discussed throughout all the article.

### B. Outline

This paper is organized as follows. In section II main principles of RESTful web Services and in section III strengths and weaknesses of RESTful web services are defined. In section IV SOAP [5] and RESTful Web Services are compared and finally in section V, we draw our conclusion.

## II. MAIN PRINCIPLES

### A. Uniform Interface

*Using http methods that follows the protocol as defined by RFC 2616 is the one of the crucial* characteristics of a RESTful web service [7]. REST design principle uses four operations these are create, read, update and delete (CRUD) and HTTP methods. POST method is used to create a resource on the server, GET is used to retrieve a resource, PUT is used in order to change or update and also DELETE is for deleting resources. HTTP methods and their verbs are shown in the figure below.

HTTP	CRUD Equivalent
GET	read
POST	create,update,delete
PUT	create,update
DELETE	delete

Fig.1

### B. Being Stateless

RESTful Web service application (or client) is designed as a stand operation. All communications must be stateless in nature. The request of clients must have all needed information in order to understand on the server and there are no restrictions between client and server via HTTP. Being stateless of communication also improves the scalability of application.

In the stateless figure is shown below each request is not depended on previous request because of the fulfilling all information alone. Also, thanks to stateless design has less complicated design, it helps gaining better performance on the application.

---

This work was supported in part by the Software Engineering Department of Bogazici University.

Baran Ipek is with the Software Engineering Department, Bogazici University, Istanbul (e-mail: baran.ipek@gmail.com).

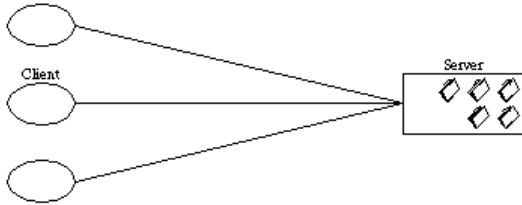


Fig.2

### C. Self-descriptive messages

Many types of resources can be used as representations. For instance HTML, XML, plain text, PDF and JPEG files are convenient for transmission[7]. Each message holds metadata information about the resource in order to interpret itself. Using MIME types allows the service can be called by variety of clients running on different platforms.

### D. Resource identification through URI

A fourth RESTful Web service characteristic is all about the URIs. URI is a kind of self-documenting interface for a software developer what it points to and to receive related resources and the structure of a URI should be predictable and understandable[1].

In order to achieve high level of usability, we can define structure like URIs. The Root and node beneath it and each node may consist several branches. We can say that “www.xyz.com” is the root, topic is the node and subs is a branch in this example”http://www.xyz.com/topic/subs”.

### E. Execution Scope

Regarding the execution scope definition, the RESTful uses the HTTP request URI. Therefore, the URI does not contain the service path only, but also any other parameters to uniquely identify the resource to be affected [15].

Five general principles are defined above and Figure 3 shows an example for a REST workflow application. All clients need to know about the remote process is the address of this process in form of a URI, which is consist of the machine name, a distinct object type and a process ID[16]. GET request sent to this URI results as an XML document that provides more information about the process and the client then sends a POST operation to the process invocation URI that contains the context data for the process instance. The process factory creates a process instance and returns the URI of the process instance to the invocator. The interaction between client and process instance is also characterized through the use of GET and POST commands, but in this case the POST command is used to manipulate the state of the process

instance, until it is completed [16].

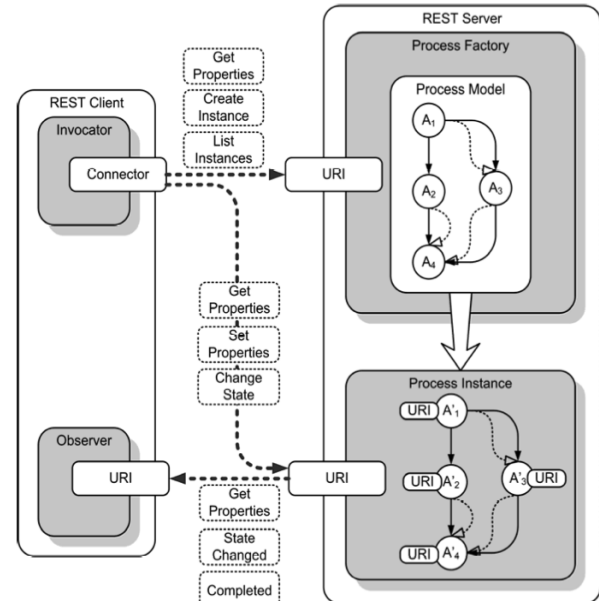


Fig.3

## III STRENGTHS AND WEAKNESSES

### A. Strengths

RESTful web services are perceived to be simple because they use well-known standards like (XML, URI, MIME etc). HTTP clients and servers are available for all major operating system/hardware platforms. Thanks to lightweight infrastructure it is inexpensive to acquire[4]. It is known how to scale a stateless RESTful web service to serve a very large number of clients, thanks to the support for caching, clustering and load balancing built into REST.

HTTP port 80 is generally is open in most firewall configurations, thanks to this feature no extra configuration is needed by default. Software developers do not make much effort to call web services, they can reach via a simple browser and begin testing it without developing client-side software. Developers do not need for additional messaging layer to get response from server. There are not many tools support RESTful web service but also this provides less reliance on tools.

In addition to this, JavaScript Object Notation (JSON [9]) supportance gives more leeway to optimize the performance of a web service. This is feature is supported by several APIs and also its implementation is very easy for developers.

The components of RESTful web service deployments are independent by each other. The content of individual web applications can be changed without adapting further sides. Independent deployment is a requirement in very large systems like the World Wide Web or the email services.

Shortly REST's client-server separation design simplifies component implementation, reduces the complexity, improves the effectiveness of performance tuning and increases the scalability of application.

### B. Weaknesses

There are still arguments regarding the commonly accepted best practices for building RESTful Web services. REST design tied to the HTTP transport model. Because of the fact that proxies and firewalls may not always allow HTTP connections that use PUT and DELETE verbs. POST and GET are the verbs that commonly used method as while transferring XHTML form[4]. Unlike POST method, GET method has size restrictions more than 4 KB as an input data. The server probably refused the request or crash may occur in case of the limit exceeding.

Due to using point-to-point communication model it is not usable for distributed computing environment where message may go through one or more publishers. The server-side application cannot run on many different servers to serve it. Because each client holds all information and stand itself to get response from server.

Another specific weakness of RESTful Web Service is concerning secure communication. There are lack of standards support for security, policy, reliable messaging. Therefore developers have to deal with security issues by extra coding or using other network tools in order to handle it. Transport-level security (TLS/SSL) may be used for secure peer-to-peer authentication, but these techniques are not adequate when requests for authentication are based on user delegation. In another words site authentication is not allowed to all users. Therefore security is gained importance nowadays among the community.

## IV RESTFUL VS SOAP WEB SERVICES

It is a hot topic for the community nowadays. REST web services share many characteristics with SOAP based web services like remote procedure call (RPC) but also differ in several important ways. It can be said that choosing RESTful or SOAP depends on your needs in general. Both of them are two different philosophies therefore before comparison, it is crucial to know the pros and cons of two. We make comparison on the three main categories, these are principles comparison, conceptual comparison and technical comparison but before debate, we need to take a glance at SOAP based web services.

In the beginning, SOAP(Simple Object Access Protocol) was announced in 1998 and it was planned as an alternative to middleware technologies like CORBA and DCOM.SOAP was gained acceptance by the W3C in 2000 and became a web service standard together with WSDL and XML Schema among the community. It is also supported by big vendor like BEA Systems, IBM,

Microsoft, and SUN that allows exchange information between in different, distributed environment.

Web Services Description Language (WSDL)is an XML language in order to define interfaces syntactically. It provides The WSDL specification provides network endpoints, or ports as an XML format for documents for this purpose [12].

Client program can read the WSDL file to decide what operations are available on the server. Several data-types are embedded in the WSDL file in the form of XML Schema. In WSDL files messages are the abstract descriptions of the data being exchanged, and port types, which are abstract collections of operations [13]. In particular, WSDL allows tools (such Apache Axis, or Sun's JAX-RPC implementation) provide you to map operations of a Web service to methods in several object-oriented programming languages. A WSDL file is shown in the figure below.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloService"
  targetNamespace="http://www.ecerami.com/wsdl/HelloService.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://www.ecerami.com/wsdl/HelloService.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <message name="SayHelloRequest">
    <part name="firstName" type="xsd:string"/>
  </message>
  <message name="SayHelloResponse">
    <part name="greeting" type="xsd:string"/>
  </message>
  <portType name="Hello_PortType">
    <operation name="sayHello">
      <input message="tns:SayHelloRequest"/>
      <output message="tns:SayHelloResponse"/>
    </operation>
  </portType>
  <binding name="Hello_Binding" type="tns:Hello_PortType">
    <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="sayHello">
      <soap:operation soapAction="sayHello"/>
      <input>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:examples:helloservice"
          use="encoded"/>
      </input>
      <output>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:examples:helloservice"
          use="encoded"/>
      </output>
    </operation>
  </binding>
</definitions>
```

Fig.4

SOAP has two main structures like many other message formats: header and body. Header part is optional but body part is mandatory. Header consists of application-specific information (like authentication, payment, etc) about the SOAP message. The body part contains the actual SOAP message. A basic SOAP request and response is shown in the figure below.

The request:

```
GET /StockPrice HTTP/1.1
Host: example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:s="http://www.example.org/stock-service">
  <env:Body>
    <s:GetStockQuote>
      <s:TickerSymbol>IBM</s:TickerSymbol>
    </s:GetStockQuote>
  </env:Body>
</env:Envelope>
```

The response:

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:s="http://www.example.org/stock-service">
  <env:Body>
    <s:GetStockQuoteResponse>
      <s:StockPrice>45.25</s:StockPrice>
    </s:GetStockQuoteResponse>
  </env:Body>
</env:Envelope>
```

Fig.5 SOAP request and response

### A. Comparison Of Principles

When it comes to compare two design models, they have different architectural principles. REST considers HTTP as an application but SOAP uses it as a transport protocol. Due to using HTTP verbs and URIs in application, it becomes a part of web but from the perspective of SOAP, HTTP is used for handshaking between applications [3]. Using HTTP as a tunneling protocol gains ability for remote communication through firewalls.

Time/availability is another important aspect of loose coupling concerns the ability for service consumers to interact with the service providers when the latter is not available. Therefore, clients are not affected when services suffer from temporary downtime. SOAP based provides using to implement message-based (as opposed to RPC-based) messages can be transferred using persistent, reliable queues. On the contrary RESTful focus on RPC-like, synchronous interactions when an HTTP server is down; they have to handle every failure themselves [3].

### B. Conceptual Comparison

SOAP and REST have a different approach to the definition of the Web service interfaces. Comparing the complexity of Web service interfaces, REST is seen as simpler in several aspects. RESTful can be called by any client or server application with HTTP support and an HTTP GET command. This interface's simplicity has important benefits over SOAP web services. Any developer can access and call easily without any extra knowledge. In contrast, SOAP requires specific XML knowledge and developers need to use SOAP toolkit in order to call service and parse the results [4].

### C. Technology Comparison

When it comes to technological comparison, standardized message format is the first issue. RESTful services do not use a single format for representing resources. It provides a variety of MIME document types which may also include SOAP itself and this multiple representation formats requires more maintenance effort than SOAP. Unlike RESTful, SOAP has a single message format.

Whereas SOAP Web services rely on a standard, interface description language is WSDL, RESTful Web services are more human-oriented approach than SOAP, and extensive documentation of the API gives facility to developers when generating web service client [3].

There are several WSDL tools that helps can generating client stub code for most programming languages, reduces the complexity of remotely interacting with a service. For RESTful Web services, developers have to manually write the code to according to the resource URIs.

Security perspective is another crucial debate while comparing both of them. In general, basic guarantees of protocols such as HTTP and HTTPS (point-to-point SSL security) are shared by both RESTful and SOAP services. Even SOAP uses remote procedure calls (RPC) [14] is a good way for security but REST followers opposed that claim because of there is no way to know whether request wants to do without looking into the SOAP envelope but In RESTful requests can be analyzed and also GET method can always be considered safe because it can only query data.

Another comparison issue is bandwidth usage. SOAP services require an XML wrapper around every request and response. Due to REST does not deal with many extra xml markup and this makes more lightweight than SOAP [11]. Since RESTful APIs can be consumed using simple GET requests proxy can cache their response very easily. On the other hand, SOAP requests use POST method and require complex XML requests to be created that makes response-caching difficult.

## IV CONCLUSION

In this paper provided deep information concerning RESTful Web Services and we made comparison in RESTful Web services and WSDL/SOAP- based web services by using three different level. On the principle level, the two approaches have similar quantitative characteristics. On the conceptual level, they have difference in interface implementation.

## SWE577 2011S

On the technology level, if you ignore more advanced features and only compare POX/HTTP and SOAP/HTTP it can be said that two approaches are two technology-level variants of the same conceptual design. But also there are some differences between them like scalability, reliability, message-level security issues that we covered throughout all the paper.

All in all, REST scores better with respect to flexibility and control, but require a lot of low-level coding. On the other hand SOAP has wider tool support and programming interface convenience even WSDL reading needs much effort. The main conclusion from our comparison is , use RESTful services for a specific subject, purpose applications over the Web and also for the business scenarios, if your scenarios are simple. You should prefer SOAP Web services in professional enterprise application integration with a longer lifespan and advanced interface.

## References

- [1] IBM, RESTful Web Services: The Basics [6 November 2008]. Available from Internet:  
<URL: <http://www.ibm.com/developerworks/webservices/library/ws-restful>>
- [2] Wikipedia, Representational State Transfer [2010]. Available from Internet:  
<URL: [http://en.wikipedia.org/wiki/Representational\\_State\\_Transfer](http://en.wikipedia.org/wiki/Representational_State_Transfer)>
- [3] R. Fielding. Architectural Styles and The Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, 2000.
- [4] Jopera, Restful Web Services vs. "Big" Web Services Available from Internet:  
<URL: <http://www.jopera.org/files/www2008-restws-pautasso-zimmermann-levmann.pdf>>
- [5] Wikipedia, Soap [2010] Available from Internet:  
<URL <http://en.wikipedia.org/wiki/SOAP>>
- [6] Available from Internet  
<URL <http://www.xfront.com/REST-Web-Services.html>>
- [7] Available from Internet  
<URL <http://www.w3.org/>>
- [8] Available from Internet  
<URL <http://www.json.org/>>
- [9] Available from Internet  
<<http://ajaxonomy.com/2008/xml/web-services-part-1-soap-vs-rest>>
- [10] Available from Internet  
<<http://geeknizer.com/rest-vs-soap-using-http-choosing-the-right-webservice-protocol/>>
- [11] Available from Internet  
<URL [http://en.wikipedia.org/wiki/Web\\_Services\\_Description\\_Language](http://en.wikipedia.org/wiki/Web_Services_Description_Language)>
- [12] Available from Internet  
<URL <http://www.w3.org/TR/wsdl>>
- [13] Available from Internet  
<URL [http://en.wikipedia.org/wiki/Remote\\_procedure\\_call](http://en.wikipedia.org/wiki/Remote_procedure_call)>
- [14] IADIS International Conference WWW/Internet 2009 SEMANTIC WEB SERVICES: A RESTFUL APPROACH  
Otávio Freitas Ferreira Filho, Maria Alice Grígas Varella Ferreira  
University of São Paulo, Polytechnic School São Paulo, Brazil
- [15] Developing Web Services Choreography Standards – The Case of REST vs. SOAP Michael zur Muehlen1a, Jeffrey V. Nickersona, Keith D. Swensonb