International Journal of Computer Science and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 5, Issue 4, Jun - Jul 2015, 39-44

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

HANDLING REST REQUESTS FOR SOAP WEBSERVICES

SAGAR GUPTA

Senior Software Developer, Amadeus Labs, Bangalore, Karnataka, India

ABSTRACT

In the current IT world, data sharing plays a vital role in business functionality. Data is shared in various formats via different protocols. One of the standard and sophisticated solutions is by use of web services. Web services provide all major features that support data handling operationsviz. data conversion, data modification, data exchange within cross domain(s). Using SOA (**S**ervice **O**riented **A**rchitecture) approach is one of the widely accepted industry standard. Web services designed using SOA approach provide features such as platform independence and universal data exchange formats.

SOAP (**S**imple **O**bject **A**ccess **P**rotocol) web services are one of the various implementations of SOA approach. However, by using SOAP architecture exposing business solutions to the client applications some is sues may arrive pertaining to data exchange. The problem here lies within the implementation standard of SOAP web services which put certain restrictions on data transmission. This paper provides simpler design solutions for handling all sorts of Client platforms using various data exchange formats.

KEYWORDS: Web Services, SOAP Web Service, REST Web Service, SOAP-REST Conversion

INTRODUCTION

SOAP architecture is a widely accepted and preferred method by Clients for exposing business solutions. Often the sebusiness processes are quite critical for an organization(s). Also, they are designed with such sophistication that platform upgrade may result in complete/partial disruption of functionality which may lead to business failure. SOAP web services address these issues quite effectively. In SOAP, these critical functional modules will perform operation without worrying about the any data conversion, software platform dependency etc. Since early 90s, SOAP webservices are providing well optimized solutions to business community for handling cross-domain and cross-platform Clients.

With the rapid changes in IT world and the evolution of multiple platforms, providing cross-platform support is a critical requirement. The conventional implementations such as SOAP pose compatibility issues for business solutions. Also, providing mobile support for business sustainability is a vital concern. Mobile devices do not provide high end support in terms of platform and configuration. SOAP requests involve high amount of XML payload which is difficult to handle considering the mobile platform. Devices with low end hardware configurations are unable to handle such high data volume transfer which is used in SOAP architecture.

Representational State Transfer popularly, known as RES Taddresses these concerns REST provides all the features provided by SOAP implementation with enhanced data handling capability and lower data transmission volume compared to SOAP implementations. Nowadays, REST architecture has become a baseline for all mobile device sowing to it slow memory footprint feature, which can handle huge volume of data in all supported data exchange formats such as

JSON, XML, text or HTML.

Due to the advantages associated with REST, most of the IT solutions prefer REST architecture. However, seamless transition from SOAP to REST is a major concern that organization(s) face due to the inherent technical and strategic challenges. This shift in architecture will require investment in terms of capital, effort and time. Also, there is no assurance if the transition will be error-free and work for all data exchange formats. This architecture provides a simple and fool-proof way for working with these two different design patterns.

The design approach described in this paper bridges the gaps exposed by using these two technology solutions and supports all data exchange formats used by these platforms.

Architecture Diagram

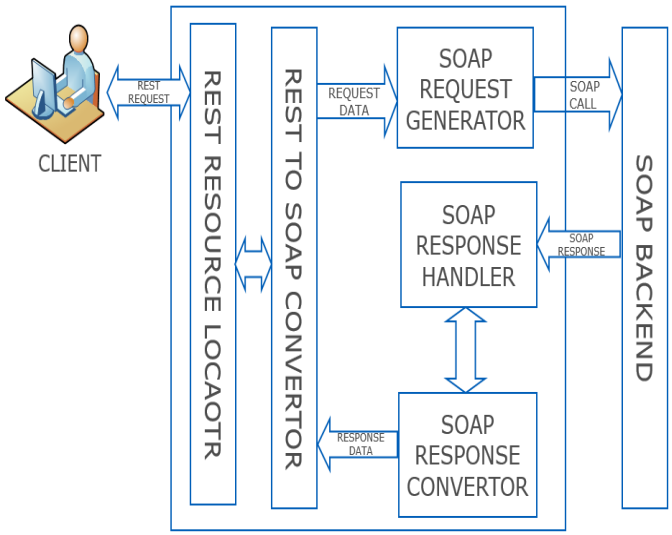

Figure 1

DISCUSSIONS

REST requests sent from Client end need to be handled by a server side technology which can parse and understand all data exchange formats used by the Client. The primary objective of this platform is to provide gateway to Clients to access SOAP backend using REST requests, which will also transform the response into requested form slike JSON, XML, text or HTML for the client.

Rest Resource Locator: This module provides the access points to client to which the REST requests should be sent in form of GET or POST request. Once the request reaches the gateway, it is passed to converter module which deals with data handling.

REST to SOAP Converter: The data sent by gateway module might be in different formats like JSON, which is not the desired request format as needed to form the SOAP request. This module takes care of all data conversions. It converts the data into standard format required for making a valid SOAP request.

SOAP Request Generator: After data conversion by the converter module, a valid request XML must be formed to call respective SOAP operation. This module takes care of creation of a valid request using the data sent, which will be validated against the schema definition required for making a valid SOAP request. This module also takes care of data

compression/encryption techniques required to send the data to respective SOAP server.

SOAP Backend: Once the request is sent by the platform the SOAP server will process the request and send back the SOAP reply. This reply is typically in form of XML which needs to be handled and parsed by converter framework to verify the various request response actions like whether an operation request resulted into technical failure, functional failure or send a valid reply. These various outcomes need to be handled and communicated to the client in standard format like SOAP faults.

SOAP Response Handler: The response retrieved from the SOAP backend can be encrypted or compressed. To parse the data for creating the REST response the encryption and compression need to be removed from the response after which data can be fed to converter module for data conversion to form valid REST reply.

SOAP Response Converter: The SOAP response received from backend is in form of XML. This XML data needs to be parsed into a standard data format from which REST response can be generated. A REST response can be of various types like JSON, plain text, HTML or XML. Hence, the module is responsible for doing the respective conversion based on the client requested output format.

Technical Implementation

The architecture discussed above can be implemented by using any high end language like JAVA, .Net, Groovy etc., which is capable of handling REST as well as SOAP request/response. The developed application has to be hosted on any application server like Web Logic, WAS, TomEE etc. For proof of concept the application is developed using JAVA and is deployed on Oracle Web Logic server.

To handle the mapping between incoming REST requests and the corresponding SOAP requests a mapping technique is required. For the discussed solution JAXB mapping is used.

The REST Resource Locator module is implemented using JAX-RS framework, in java which provides the support for XML to JAVA object mapping. It also provides support for marshalling and un-marshalling of XML file against valid JAVA file definition. The gateway which is created for the clients is in the form of REST resource to which client sends the data in the form of GET or POST requests, etc.

After receiving the request from the client, the data is fed to the *REST to SOAP Converter* module. This module takes care of extracting data from the payload, like in case of POST request the payload must be parsed, or in case of GET request the data must be collected from path or query parameters. Once the data is collected the respective SOAP request is generated based on the requested operation.

The *SOAP Request Generator* module is next in-line of operation, which is responsible for recognizing the SOAP operation for which request must be generated. After identifying the operation a request is created by using JAXB or any other XML marshalling framework which can create and validate XML request. The module also takes care of various other features involved in the SOAP request generation, such as:

- **Security:** The SOAP backend accepts only secure channel communication. The request must consist of various security features like TLS (**T**ransport **L**ayer **S**ecurity) or WSS (**W**eb **S**ervices **S**ecurity) etc. Handling such security features is possible in JAVA using JAX-WS security framework.

- **Compression:** The data sent as body of SOAP request might be heavy in memory footprints. Transmission of such high volume data over the wire might add some lag or increase network traffic load. To overcome such problems, compression strategies like GZIP can be used.
- **Attachments:** Some web service accept attachments or binary data also as a valid request. For handling such scenarios various strategies like MTOM (**M**essage **T**ransmission **O**ptimization **M**echanism) should be implemented to create or parse the binary data.

Once the message is received from SOAP backend, the *SOAP Response Module* will process the response by using any un-marshalling libraries like Aegis or JAXB. The un-marshalled JAVA objects can be used to retrieve the data to form the valid REST response.

In case output data format is JSON or HTML, data retrieved from the SOAP web service must be handled and processed in a manner that resultant data represents the data type as requested by the client. This conversion of SOAP response into client's requested format is handled by *SOAP Response Converter*.

JSON format is one of the most commonly used data sharing format. To convert the data into valid JSON format a valid JSON structure has to be created by the framework which should be in sync with the format handled at the client side. JSON objects can be created in JAVA which can be sent as REST response in form of plain string or JSON format to the client. For handling JSON format, specific data handling techniques must be used by framework so that the transmission obeys all the principles of HTTP protocol and standards set by REST.

Handling REST resources in web technology poses certain risks which might even lead to data leakage to unknown sources at client's end. Some of these attacks are:

- **CSRF** (**C**ross-**S**ite **R**equest **F**orgery): In this attack while handling REST requests, client performs some malicious actions on the trusted site because of un-secure content received by the REST response. This can be avoided by using token based mechanism in which every request to server will get validated by unique token sent by client and vice versa.
- **XSS** (**C**ross **S**ite **S**cripting): This attack is more prominent when dealing with HTML response. If some malicious script tag is added to the HTML response sent from the unknown source, sensitive client information can be leaked. To handle such scenarios proper data encoding schemes like UTF-8, URI must be implemented such that any malicious script will not get executed without client's permission.

Hence, for ensuring a secure transmission of data between the REST client and framework some sort of security mechanisms must be implemented

Benefits

- The solution detailed above bridges the gap between two different backend.
- The solution will provide more flexible approach to open SOAP based backend, as irt provides customized data handling and data parsing mechanisms.
- No need to migrate whole business logic to the new REST platform because the same back-end can handle REST as well as SOAP requests simultaneously.

- The approach is highly scalable as the SOAP services which are handled by the framework can be enabled/disabled with less manual intervention.
- Handles all request and response format structures. SOAP web services provide support only for XML formats. However, with this approach all data formats are supported.
- This offers secured data transmission and conversion.

CONCLUSIONS

With the growth in hardware and software technologies overtime, it is essential to design a framework or an approach that can adapt quickly to technological changes and provides seamless support to all platforms without affecting the core implementation.

The framework outlined in this paper answers all these concerns and provides a sophisticated solution for handling SOAP backend for REST Clients, without any investment of infrastructural cost and minimum effort.

REFERENCES

1. http://www.oracle.com/technetwork/articles/javase/index-140168.html.
2. https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF).
3. https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).
4. https://www.w3.org/TR/soap.
5. https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

AUTHORS DETAILS

Sagar is a Senior Software Developer with Amadeus Labs based out of Bangalore, Karnataka. He is a certified cloud computing architect and database solution advisor with over 5 years of experience in solution designing and architectural review. He had published many papers in international journals on cross-domain communication and interoperability for cutting edge solutions. You can find Sagar on

LinkedIn: https://www.linkedin.com/in/ersagargupta Or Academia: https://independent.academia.edu/SagarGupta7