Study of test script design methods for Web Service performance testing

Peng Xu
Software Quality Engineering Research Center, China Electronic Product Reliability and Environmental Testing Research Institute, Guangzhou 510610, P.R. China
cepreixp@126.com

Abstract. Web Service interface technology is more and more widely applied in information system. And more requirements of Web Service performance testing are demanded. Nevertheless, designing the test script for performance testing is hard to take into practice. In this paper, two kinds of test script design methods for Web Service performance testing are presented by using LoadRunner and SOAP UI tools. That is The Service Call Method and The SOAP Method.

1 Introduction
Web Service is an API or application that can be called by the Internet. Users can call the Web Service application over the Internet, and then the application returns the data that users need. Web Service interface is mainly based on three kinds of technologies, including XML (Extensible Mark-up Language), SOAP (Simple Object Access Protocol) and WSDL (Web Services Description Language)\(^1\). Among them, XML represents the basic format of data in Web Service, while SOAP provides a standard RPC (Remote Procedure Call Protocol) method to call the Web Service, and WSDL is a language used for describing Web Service, its operation, parameters and returned value on the basis of XML\(^1\).

Web Service technology enables different applications which may run on different machines and base on different language, platform, and internal protocol to exchange data with one another, without the help of additional or specialized third-party software or hardware\(^1\). Given these advantages, Web Service interface technology has found an increasingly wide utilization in current information systems\(^1\).

With the extensive utilization of Web Service technology, More and more softwares based on web service technologies are developed. Before their releases on the Internet, it is necessary to evaluate these systems' performance, especially their response time under different workload pressures\(^6\)\(^7\). At present, tools for Web Service performance testing mainly include LoadRunner\(^2\), SOUPUI\(^4\), JMeter\(^5\), etc\(^1\), among which LoadRunner is the mainstream one\(^2\).

The processes of software performance testing using LoadRunner can be divided into three steps\(^2\)\(^3\), in which the first step is editing and debugging test scripts, the second is designing the test scenarios and executing test, and the third step is analysing the test results. Among the three steps, the first step is the most important and the most difficult to realize in the software performance testing using LoadRunner\(^2\)\(^3\). Of course, Web Service performance testing is no exception.
This paper provides two kinds of test script design methods which are used in Web Service performance testing. The first type design method is realized by using the `web_service_call` function in LoadRunner testing tool \(^2\[^3\]\), which is called The Service Call Method. And the second type design method can be put into practice by importing the SOAP request file of the Web Service to be tested into the LoadRunner testing tool, which is named The SOAP Method. In the end, two kinds of test script design methods are shown by taking a Web Service performance testing as an example.

**2 Two kinds of test script design methods**

This section will introduce the two test script design methods of Web Service performance testing, namely, The Service Call Method and The SOAP Method.

**2.1 The Service Call Method**

The Service Call Method is based primarily on LoadRunner\(^1\[^1\]\) which is a performance testing tool to realize. Its design processes mainly include the following parts.

The first step is analysing the input/output parameters information of Web Service to be tested. And this work could be achieved by using LoadRunner\(^1\[^1\]\). Firstly, a test script framework based on Web Service protocol should be created, and then import the WSDL address of the Web Service under test into LoadRunner. Finally, by using LoadRunner to analyse the Web Service under test, we can obtain the input/output parameters information of the Web Service, which includes parameters names, data types, etc.

The second step is applying The `web_service_call` function \(^2\[^3\]\) which is provided to be use in LoadRunner to call the Web Service under test on the basis of analysing the input/output parameters information. These can be finished by the following steps, that is, sending the request which include legal parameters to Web Service by using the `web_service_call` function, and saving the parameter values returned by the Web Service by using the `web_service_call` function after Web Service giving responses to the request.

The third step is debugging the test script to verify the correctness of the saved parameter values returned by the Web Service, and then completing the designing of the test script if they are correct.

**2.2 The SOAP Method**

By using the SOAP UI\(^4\) testing tool and the LoadRunner 11 testing tool, the SOAP method can be taken into practice. Its design processes mainly include the following steps.

The first step is to obtain the SOAP request file of Web Service under test by utilizing the SOAP UI testing tool\(^4\).

First, create a project in the SOAP UI test tool, import the WSDL address of Web Service into the project and create a Web Service request framework. In this case, the SOAP UI will analyse the Web Service under test and provide the users interface which include the request and response interface of the Web Service. Then, send request containing legal parameter values to the Web Service in the request interface and check the correctness of the response in the response interface. If it is correct, you can export the request in XML format, namely, the SOAP request file of Web Service has been got.

The second step is using The `soap_request` function which is provided to be use in LoadRunner to call the Web Service under test\(^2\[^3\]\). Firstly, a test script framework based on Web Service protocol should be created, and then use the `soap_request` function to call the SOAP request file which is got in the preceding step to make request of the Web Service.

The third step is to debug the test script of the Web Service. Base on the preceding step, send the SOAP request file to the Web Service using the `soap_request` function in LoadRunner, and then the Web Service will return the SOAP response datagram in XML format. Judge the correctness of the output parameters values in the SOAP response datagram to verdict the correctness of the SOAP response datagram. If it is correct, debugging the test script of the Web Service is finished.
3 Test script designing instances based on the two methods mentioned above

This section will illustrate the process of the two test script design methods mentioned above by taking a real Web Service performance testing as an example, and which will verify the validity of the two methods.

3.1 Introduction of the Web Service to be tested

As is shown in Table.1 below, the detailed information of the Web Service to be tested is given, which includes the entry address and the input/output parameters information of the getItemByTitle interface which is one of the interfaces of the Web Service. The entry address consists of the WSDL Address and the Server Address, by which users can call the Web Service. After sending the title of a book to the getItemByTitle interface, the interface will allow user to get the detail information of the book including book id, quantity in stock, price, ISBN, publication date, authors, publisher, etc.

Table.1 Information of the Web Service to be tested

| Parameters   | Description                                      |
|--------------|--------------------------------------------------|
| Input        |                                                  |
| title        | Data type: string, title of the book              |
| id           | Data type: int, id number                        |
| title        | Data type: string, title of the book              |
| quantity_in_stock | Data type: int, quantity in stock          |
| price        | Data type: float, price                          |
| isbn         | Data type: string, ISBN                          |
| Output       |                                                  |
| id           | Data type: int, id number                        |
| title        | Data type: string, title of the book              |
| quantity_in_stock | Data type: int, quantity in stock          |
| price        | Data type: float, price                          |
| isbn         | Data type: string, ISBN                          |
| Publication_date | Data type: Date, publication date               |
| description  | Data type: string, description of the book        |
| authors      | Data type: string, author                        |
| publisher    | Data type: string, press                         |

3.2 Test script design instance of the Service Call method

Figure.1 the Input/output parameters information of the getItemByTitle interface
When using the Service Call method to design test script, the first step is analysing the input/output parameters information of Web Service to be tested by utilizing LoadRunner11.

As is shown in Figure 1, LoadRunner11 returns the detailed input/output parameters information of the getItemByTitle interface. It’s thus clear that the input/output parameters information shown in Figure 1 is consistent with the information shown in Table 1.

After the analysis, the web_service_call function is applied to call the Web Service. Assign the book name “Linux Administration Handbook” to the input parameter “titlekeyword” as the interface input request, store the value of the output parameter “id” in the parameter “Param_id”, and debug the script to check the correctness of the output parameters values in the debugging log.

Test script written by using the web_service_call function and its debugging result are shown in Figure 2. As can be referred from the debugging log in Figure 2, Web Service correctly returned the book id number corresponding to which the book name is “Linux Administration Handbook”, namely, “4”.

With this result, the input value “Linux Administration Handbook”, which has been assigned to the parameter “titlekeyword”, can be parameterized and then get an optimized script.

3.3 Test script design instance of the SOAP Method

Using the SOAP Method to design the test script, the SOAP request file of Web Service under test should be got by utilizing the SOAP UI testing tool at first.
Send the request datagram containing legal parameters to the Web Service in the request interface providing by the SOAP UI test tool, and judge the correctness of the response. Figure 3 shows the request and response of getItemByTitle interface in the SOAP UI test tool, in which the input parameter “titlekeyword” is assigned the value “Linux Administration Handbook” and the interface give the correct response which can be proved by the output parameter “id” value, namely, “4”.

In this case, export the request which can be seen in the request interface in the SOAP UI in XML format to get the SOAP request file. And Table 2 below show the SOAP request file got by the proceeding steps.

Table 2 the SOAP request file of the Web Service

```xml
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:stor="http://www.parasoft.com/wsdl/store-01/">
  <soapenv:Header/>
  <soapenv:Body>
    <stor:getItemByTitle>
      <stor:titleKeyword>Linux Administration Handbook</stor:titleKeyword>
    </stor:getItemByTitle>
  </soapenv:Body>
</soapenv:Envelope>
```

When the SOAP request file is obtained, use the soap_request function to call the SOAP request file in the test script framework provided by LoadRunner, and consequently generate the test script of the Web Service by the SOAP Method.

Debug the test script in LoadRunner and check the correctness of the Web Service’s response in the debugging log window of LoadRunner. In Figure 4, the test script designed by the SOAP Method and its debugging result are shown, from which the conclusion that the Web Service has returned the correct response message can be made, namely, we have get a useful test script. And as well, the input parameter value “Linux Administration Handbook” which is assigned to the parameter “title” can be parameterized and thus generated an optimized script.
4 Conclusion

This paper presents two kinds of test script design methods for Web Service performance testing by using LoadRunner and SOAP UI test tools, which are The Service Call Method and the SOAP method. And then taking a Web Service performance testing as an example, the two kinds of test script design methods are demonstrated in this paper. And simultaneously, the validity of the test scripts is verified through the debugging results of each script.

References

[1] Ning Gu, Jiamao Liu, Xiaolu Chai. “The principle of Web Services and it’s research and development practice”, BeiJing, China Machine Press, 2006.
[2] Yong Yu, “Proficient in software testing and practices of using LoadRunner”, BeiJing, Posts & Telecom Press, 2010.
[3] HP Company. “HP LoadRunner Software 11.00 User Guide for the Windows operating system”, September 2010.
[4] Kankanamge, Charitha. “Web services testing with soapUI.” Packt Publishing Ltd, 2012.
[5] You Jing, Zhang Lan, Wang Hongyuan. “JMeter-based Aging Simulation of Computing System[C]” Proceedings of 2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering. Washington D. C., USA: IEEE Press, 2010:282-285.
[6] Jingmin Xie, Xiaojun Ye, Bin Li, Feng Xie, “A Configurable Web Service Performance Testing Framework”, Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications, pp.312-319, September 25-27, 2008.
[7] Sha F, Yu K, Zhang L, et al. "A performance evaluation method and it's implementation for web service." Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on. IEEE, 2010.