Implementation of a non intrusive software component to monitor web applications in a public entity

Abstract

The importance of measuring the response time of web applications is transcendental for decision-making and/or corrections on the part of public officials involved in the departments of Information systems and technologies. This research provides the conceptual framework for the development of a component J2EE (Java 2 Enterprise Edition) non-intrusive through Web Servlet filters and Interceptors of Spring Framework. It also uses the communication protocol TCP/IP for sending frames in JSON format (JavaScript Object Notation) and its subsequent reception for a stand-alone component for the record in a database. Finally, a hypothesis test between Log4j and Socket Simple technologies is performed, achieving an average record time of the plots approximately 0.05 seconds, and report that there are no statistical evidences to affirm that one is better and faster than the other.

Keywords: component, filters, interceptor, log4j, socket

INTRODUCTION

Today the need for measuring, assessing and monitoring the services provided not only by companies but also by government bodies, is of high significance in achieving corporate goals. Furthermore, in the fields of technology and information systems there are a series of mandatory regulations, as well as a set of best practices and frameworks that allow an adequate work performance and strategic alignment of the technologic area as a support to the institutional strategic plan (Peruvian Technical Standard-NTP, ISO, SOX, COBIT, COSO, etc.).

In this regard, the need to measure response times or service processing is crucial as evidence of the current situation of the organization and thus determines whether or not they meet the goals previously established by senior management. In the specific case of information technology, measuring response times of the applications or services offered via the Web, both to citizens and internal users, is an important source of data and information to make decisions, perform corrections and develops management plans of the specialized areas.

Additionally, a well-known framework like COBIT® suggests for instance the use of the Balanced Scorecard for Information Technology because it "defines clear goals and the effect of the impact of IT also demonstrates and communicates to senior management effectiveness and delivering business value" [10]. In other words, the information obtained from a measurement to quantify the real contribution of the information technology area to the entity and obtain a first indicator of the satisfaction level of service offered to users.

Therefore, this article illustrates the case study of the implementation of a component with the library type that adds in a non-intrusive way to monitor web applications. Its main target is to measure the initial and final time of a service, and then throw a message in the form of JSON (JavaScript Object Notation) a frame to a remote server that contains a stand-alone component responsible for receiving this message, interpret and record in a statistical
database. As an encoding solution arises in J2EE (Java 2 Enterprise Edition) web filter for the case of Servlets and Interceptor by AOP (Aspect Oriented Programming) for the case of Spring Framework. Finally, sending JSON messages is done by log4j, a technology that allows asynchronous sending frames as “log” using TCP/IP sockets. It also evaluates the performance of simple socket and log4j through hypothesis testing by first calculating the sample size of n=599, then based on these samples yields an average transaction log approximately a twentieth of a second (\( \mu \approx 0.05 \text{ seconds} \)) and a significance level of \( \alpha=0.05 \) it is concluded that there is not statistic evidence to affirm that superiority in performing on the part of log4j about simple.

2. WEB FILTERS AND INTERCEPTORS WITH AOP

The concept of filters appears from the 2.3 Servlet specifications. In technical terms it is established that a filter “is an object that intercepts a message from a data source and a recipient” [1]. That is, the filter performs an interception when it is invoked a procedure or method. In the development of web applications, a filter is a component that resides on the web server by filtering the requests and responses are passed between the client and a resource. Additionally, in the Java application development allows implementation of multiple filters which are linked and executed in a chain way. Figure 2 shows a graphic description of the application of filters.

In parallel, Spring Framework has its own implementation to conduct interceptions of calls that are called “interceptors”. As noted by H. Harrop [8] in his book, the basis of these interceptors is given by the paradigm of aspect-oriented programming (AOP).

As seen in figure 3, AOP presents a slight variation regarding the traditional paradigm of object-oriented programming (OOP). When a call is made to the technical services from applications, then it is applied an “aspect” in charge of intercepting different moments of the called to the service requesting for example the official moment and the last one of an invocation to an operation of a Java class.

---

1. DEFINITION OF THE SOFTWARE COMPONENT

According to Sommerville I. [14], software engineering is applied to component-based approach in the definition, implementation and composition of loosely coupled independent components within the systems. Flexible coupling means to invoke the feature via software component interfaces calls. By independent component is interpreted to be capable of being deployed in a distributed manner without the need to call external services. Here figure 1 shows a component software diagrammed with UML (Unified Modeling Language) notation.

On the other hand, among the select group of authors J2EE development experts, Dereck C. [4] defines software components as “generic utility classes and static to be used by many applications”, also provides the necessary guidelines to generate high quality components themselves. It recommends that should be a short development time and effort, maintenance short expectation that are useful in many applications because they are generic as well as a reliable availability and stability.

**Figure 1.** Graphical specification of a software component.

**Source:** Taken from Sommerville I. [14]
Nowadays, there are no companies or organizations that do not implement the use of logs to record the register of events and/or failures occurred in their enterprise applications deployed in production. Commonly, these logs of information are stored in files that are generated and automatically renaming every day. However, not only can be stored and streamed to files, but also in libraries “logs” that provide other forms of storage and transmission to a JMS queue (Java Message Service), console, email, telnet and sockets. In the case of J2EE, the most used library in public institutions and companies is **log4j.jar**. Then the Figure 4 is shown the lines of classic configuration for the use of a log through a socket.
Gupta S. [5] mentions in his publication that “the transfer of information to a log from one machine to another is a scenario of distributed storage”. In other words, log4j allows to send frames to a remote server using the protocol TCP/IP.

4. JSON FORMAT FOR SENDING FRAMES

According to the findings by N. Zakas [15], JavaScript Object Notation (JSON) was technically specified as IETF RFC 4627 in 2006 by Douglas Crockford, although it had previously been used since 2001. It is noteworthy that JSON is not a programming language or a software development technology, but is a standard format for sending and receiving messages in plain text. Below in figure 5 it is shown an example of the JSON format for the proposed solution in this article:

```json
{  "a" : 5,
   "b" : "2013-03-26 02:43:51.804",
   "c" : "2013-03-26 02:43:52.016",
   "d" : "10428489069FSANTOSL",
   "e" : "saveOperation",
   "f" : "{"num_opel" : 45034008}"
}
```

**Figure 5. JSON example**

*Source: Own elaboration*

It is noted that the text in square brackets is framed both the beginning and end of the message. Fields that are sent are described in double quotes, followed by a colon and the value to send. Each group of fields with their respective values is separated by commas.

5. SOLUTION PROPOSED FOR THE IMPLEMENTATION

Applying the concepts explained in the previous sections, we proceed to give solution to implementing a non-intrusive software component to monitor web applications.

The following items must be considered for a successful implementation:

- Lines below are stated the properties required by the socket appender as the remote host’s IP number **192.168.1.116** and the listening port **9010** on that server. It was chosen the most stable communication protocol TCP/IP (Transmission Control Protocol / Internet Protocol) that “belongs to the transport layer/network which were developed by the U.S. Department of Defense’s Advanced Research Projects Agency Network (ARPANET) by Vinton Cerf and Bob Khan in 1974” [3].

- As noted by Gustafson D. [6], the design is “artistic or creative part of the software development process.” This process converts the “what” into “how”, i.e. understanding and documenting software requirements. For this research are developed in figures 6 and 7 the design class diagrams that provide support for coding level solution.

- To design class diagrams it is used the UML (Unified Modeling Language), which allows “bind class static content through relationships. Also shows the variables and member functions of the class” [2]. In figure 6 it is defined the class “BeanTimer” with the following fields:
  - **A** ➔ Service code
  - **B** ➔ Start date of the process
  - **C** ➔ End date of the process
  - **D** ➔ Transaction user
  - **E** ➔ Name of the method or monitored operation
  - **F** ➔ Field of 255 characters to store any chain
  - **G** ➔ Indicator if the process was conducted successfully or not

- An example of BeanTimer using JSON is displayed in Figure 5. Letters are used for this bean because these data travels through the network and is required to occupy the fewest possible bits thus are used as short descriptions letters.

- On the other hand, Figure 7 shows the classes necessary to build stand-alone component responsible for receiving the frames and record on a relational database. The class diagram ListenerLog4j that by operating handles go **recibirMensaje** receiving frames them is coming to this component. It uses an infinite loop to achieve this purpose. Then ListenerLog4j creates what in the Java programming language is called “Thread”.

---

**Figure 6. Design class diagrams for BeanTimer**

*Source: Own elaboration*
Quoting the statement by Sierra K. [13], a Thread is divided into two parts: On the one hand is defined as any object in Java that has variables and methods, and has a beginning and end within the pile of computer memory. On the other hand, there is the so-called thread of execution is an individual process that has its own call stack. Therefore, due to the high turnout of web applications must implement the record to the database using Threads. In this way, Figure 7 shows `ServerThreadLog4j` class that implements the Runnable to implement a Thread.

In the Figure 8 it is shown the Java code for shipping `BeanTimer` class instance to the remote server. On the left side implementation is illustrated by Simple Socket should indicate where in the IP source code and listen port of destination, as well as opening and closing a connection to the socket. However, on the right side displays Log4j implementation by requiring only use `log.debug` function, because the library itself is responsible for opening and closing connections, as well as use the file of configuration shown in the Figure 4.
− In figure 9 is illustrated the non-intrusive deployment configuration XML files developed component and must be added to all the applications to be monitored. On the left side is the configuration using servlets and filters for the configuration right for Spring AOP Framework.

− Finally, the company’s technology architecture is diagrammed in figure 10. Here are displayed calls from browsers Internet, Extranet and Intranet applications. Then a distributed server Oracle Web Logic® application receives requests for that at that moment the component intercepts the initial and final time of the process. It also shows the JSON sending messages to the remote server.

6. STATISTICAL SAMPLING

Due to the high concurrency, for implementing web applications should not be ignore the so-called stress tests to analyze and verify optimal implementation of service monitoring component. Available on the market a number of tools that provide interfaces for stress testing, Jmeter® being the country’s leading proceeds to use this tool with detailed usage can be found in the book of Halili E [7]. Additionally, the area of technology infrastructure of the public entity under study delivers an estimated 200 000 daily records for this component. Applying the concepts of classical statistics proceed to calculate the sample size for a proper study of the processing time. To do this, says Hayter A. [9], the sample size for a universe of 200 000, a significance level of 0.05 \( (Z_{0.05} = 1.96) \), an estimate of \( p \) unknown for which one must assume a value of 0.5 and erroneous sample percentage not exceeding 4% \( (E = 0.04) \) is calculated by:

\[
n = \frac{Z_{0.05}^2 \hat{p} \hat{q} N}{E^2 + \frac{Z_{0.05}^2 \hat{p} \hat{q}}{2}} = \frac{1.96^2 (0.5)(0.5)(200 000)}{200 000 (0.04^2) + 1.96^2 (0.5)(0.5)} = \frac{192 080}{598.4539} \approx 320.9604
\]

With the sample size of 599 proceeds to obtain that size of socket simple and log4j, using Jmeter®, as shown in Table 1. Furthermore, the arithmetic mean and standard deviation are calculated for each sample.
Due to the high concurrency, for implementing web applications should not be ignored the so-called stress tests to analyze and verify optimal implementation of service monitoring component. Available on the market a number of tools that provide interfaces for stress testing, Jmeter® being the country's leading proceeds to use this tool with detailed usage can be found in the book of Halili E [7]. Additionally, the area of technology infrastructure of the public entity under study delivers an estimated 200,000 daily records for this component.

Applying the concepts of classical statistics proceed to calculate the sample size for a proper study of the processing time. To do this, says Hayter A. [9], the sample size for a universe of 200,000, a significance level of 0.05, an estimate of p unknown for which one must assume a value of 0.5 and erroneous sample percentage not exceeding 4% is calculated by:

$$n = \frac{Z_{\alpha/2}^2 \cdot \hat{p} \cdot (1 - \hat{p})}{E^2}$$

With the sample size of 599 proceeds to obtain that size of socket simple and log4j, using Jmeter®, as shown in Table 1. Furthermore, the arithmetic mean and standard deviation are calculated for each sample.

Table 1: Monitoring processing times (shows the four first and last)

<table>
<thead>
<tr>
<th>Test Log4j (599 samples)</th>
<th>Simple Socket Test (599 samples)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register date</td>
<td>End date service</td>
</tr>
<tr>
<td>2013-03-29 15:00:07.328</td>
<td>2013-03-29 15:00:07.259</td>
</tr>
<tr>
<td>2013-03-29 15:00:12.299</td>
<td>2013-03-29 15:00:12.252</td>
</tr>
<tr>
<td>2013-03-29 15:00:17.081</td>
<td>2013-03-29 15:00:17.041</td>
</tr>
<tr>
<td>2013-03-29 15:00:19.73</td>
<td>2013-03-29 15:00:19.671</td>
</tr>
<tr>
<td>……………</td>
<td>……………</td>
</tr>
<tr>
<td>2013-03-29 15:25:15.319</td>
<td>2013-03-29 15:25:15.276</td>
</tr>
<tr>
<td>2013-03-29 15:25:18.953</td>
<td>2013-03-29 15:25:18.902</td>
</tr>
</tbody>
</table>

$$\bar{x}_1 = \frac{\sum x_j \cdot \sum x_j}{n \cdot 599} = \frac{31.382}{599} = 0.05239$$

$$\sigma_1 = \sqrt{\frac{1}{n-1} \sum (x_j - \bar{x}_1)^2}$$

$$\bar{x}_2 = \frac{\sum x_m \cdot 599}{599} \cdot \frac{599}{599} = 30.04 \cdot 0.05014$$

$$\sigma_2 = \sqrt{\frac{1}{n-1} \sum (x_m - \bar{x}_2)^2}$$

$$\sigma_1 = \sqrt{\frac{1}{599-1} \sum (x_j - 0.05239)^2} = 0.011981735$$

Source: Own elaboration.
In the Diff column provided in table 1 is expressed in seconds the result of the subtraction of the register date and the end date service. This provides the processing time since the network sends the plot to its inclusion in the Informix® database, and socket both log4j simple. In both cases it gives an average of approximately 0.05 seconds and a standard deviation of about 0.011.

The figure 11 appreciates a diagram of linear dispersion sample for log4j. It is shown that the values are close to 0.05 seconds and the vast majority below 0.07 which is the maximum tolerated by the public entity of this study.

**HYPOTHESIS TEST**

Based on the data obtained from the samples of the previous point, it proceeds to calculate success rates as shown in Table 2. Additionally, account and considered a successful time for those less than 0.07 seconds.

![Figure 11. Diagram of the times for the samples of Log4j.](source: Own elaboration with Microsoft Excel®)

<table>
<thead>
<tr>
<th>Table 2. Times of monitoring processing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Technology</strong></td>
</tr>
<tr>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Less than or equal times to 0.07 seconds</td>
</tr>
<tr>
<td>Total number of measured times</td>
</tr>
<tr>
<td>Success percentage</td>
</tr>
</tbody>
</table>

Source: Own elaboration.

With the values of tables 1 and 2, and assuming that this process behaves under a normal distribution statistical calculations are made for achieving determine if one technology is better than the other.

**Step 1:** It is defined the sample of log4j as $p_1$ and the one of the socket simple as $p_2$.

**Step 2:** The assertion of a higher speed of log4j is expressed as $p_1 > p_2$.

**Step 3:** If $p_1 > p_2$ is false, then $p_1 \leq p_2$. The hypothesis are defined as: $H_3: p_1 = p_2$ (Null hypothesis)

$H_1: p_1 > p_2$ (Alternative hypothesis)

**Step 4:** The significance level is established as $\alpha = 0.05$.

**Step 5:** Apply the normal distribution and calculate the estimate of the grouped sample $\bar{p}$.

$$\bar{p} = \frac{x_1 + x_2}{n_1 + n_2} = \frac{576 + 575}{599 + 599} = 0.96$$

Where: $\bar{q} = 1 - 0.96 = 0.04$

**Step 6:** Calculate the test statistic value expressed by the following equation:
\[
z = \frac{(\hat{p}_1 - \hat{p}_2) - (p_1 - p_2)}{\sqrt{\frac{\hat{p}\hat{q}}{n} + \frac{\hat{p}\hat{q}}{n}}} = \frac{(0.576 - 0.575) - 0}{\sqrt{\frac{(0.96)(0.04)}{599} + \frac{(0.96)(0.04)}{599}}} = 0.00166945 \approx 0.14743683
\]

As this is a normal distribution, then you get left area for the \( z \) statistic (see table in [9]), being 0.5557. Therefore, the value \( P \) (area to the right):

\[ P = 1 - 0.5557 = 0.4443 \]

**Step 7:** Since \( P \) is not less than the significance level (\( P > \alpha \)), then the null hypothesis \( H_0 \) is not rejected.

8. CONCLUSIONS AND RECOMMENDATION

− According to the results obtained in the sampling and testing of hypotheses, we conclude that in both cases both log4j and simple socket means and standard deviations are very similar and with little variation. In addition, by not rejecting the null hypothesis \( H_0 : p_1 = p_2 \), can not be said to be more efficient log4j logging response times than simple socket.

− The design using the UML modeling language and the implementation of good practices in software engineering allow successfully develop and implement non-intrusive component changes in the monitored applications just modifying XML files.

− As discussed throughout this article, it is concluded that the log4j implementation is the best choice because using fewer lines of code for sending frames; you configure the IP address and destination port to an external file the enterprise applications, making these parameters are subsequently modified without altering the source code. That is superior in terms of quality.

− It is highly recommendable to install the software component deployed as a library and distributed within the application server instead of including them in the “classpath” of each web application. As a result, the component is achieved centralize allowing quick and efficient updates. For this research were performed configurations and facilities suggested by Patrick R [11] in the case of Oracle Web Logic®, see pages 286, 287 and 288 for details. For other distributions should consult their own documentation.

9. BIBLIOGRAPHY


