ISSN 2250-3153

Monitoring and Notifications Solution for Web Server (MANOWS)

Pratyush Poddar[1], Praveen Kumar[2]

[1] B.Tech (CS&E), Amity University, Noida
[2] Assistant Professor, Amity University, Noida

Abstract- At Manows, it is expected that as a result of the increasing complexity of the internet and the intensifying e-competition, web-servers availability and smooth functionality will become an even more important competitive advantage for its users. Manows will check target web-server(s) continuously in a pre-defined interval for availability and send notifications if it finds any exceptional scenario. A very flexible configuration strategy will be adopted in Manows so that the web-servers to be monitored, notification methodology and alert recipients can be configured very easily and efficiently in it. The project will be developed with open standards and J2EE, a platform independent technology; its users therefore need not bother about the operating system where it will be running.

Index Terms- web-server, notifications, Website monitoring

I. INTRODUCTION

Monitoring and notification solutions of web server (MANOWS) refers to automatic detection of changes made to World Wide Web pages and notification to interested users by email or other means. Whereas search engines are designed to find web pages, Change Detection and Notification (CDN) systems are designed to monitor changes to web pages. Before change detection and notification, it was necessary for users to manually check for web page changes, either by revisiting web sites or periodically searching again. Efficient and effective change detection and notification is hampered by the fact that most servers do not accurately track content changes through Last-Modified or ETag headers.

Several web browsers have extensions or options that let people know when e-mail is waiting in one or more of their inboxes. Other e-mail notification programs run independently and display a message or icon when e-mail arrives E-mail notification programs are very similar to the clients commonly used to send and receive e-mail. The difference is that e-mail notification applications are written specifically to check and report the performance of the web-server. Some programs allow people to read and even answer their mail, and others simply display a link that connects people to a web mail interface. In general, however, e-mail notification programs don't have the ability to save mail to disk or to organize messages into folders.

Web-server monitoring is the process of testing and verifying that end-users can interact with a website or web application. Website monitoring is often used by businesses to ensure that their sites are live and responding.

Website monitoring companies that offer website performance monitoring allow businesses to simulate the actions of thousands of visitors to a website and observe how it responds. They also simulate visitors across multiple geographies and servers Internet connections. Performance monitoring tools send out alerts when pages or parts of a website malfunction, which allows the webmaster to correct issues faster.

Website Security monitoring is also used to verify that a domain (and web site) is not only responding properly, but has not been hacked, blacklisted or hijacked. Website monitoring can be done from both inside and outside of a corporate firewall. Traditional Network Management solutions focus on inside the firewall monitoring, whereas external performance monitoring will test and monitor performance issues across the Internet backbone and in some cases all the way to the end-user.

Inside firewall monitoring is done by special hardware appliances which can help you determine if your internal applications' sluggish performance is caused by: design of applications, internal infrastructure, internal applications or connections to any public internet. External performance monitoring is also known as end-user monitoring or end-to-end performance monitoring.

II. ABOUT XML

XML (eXtensible Markup Language) is a meta-language; that is, it is a language in which other languages are created. In XML, data is "marked up" with tags, similar to HTML tags. In fact, the latest version of HTML, called XHTML, is an XML-based language, which means that XHTML follows the syntax rules of XML.

XML is used to store data or information. This data might be intended to be by read by people or by machines. It can be highly structured data such as data typically stored in databases or spreadsheets, or loosely structured data, such as data stored in letters or manuals.

An XML document is made up of the following parts.

- An optional prolog.
- A document element, usually containing nested elements.
- Optional comments or processing instructions.

III. THE PROLOG

The prolog of an XML document can contain the following items.

- An XML declaration
- Processing instructions
- Comments
- A Document Type Declaration

The XML Declaration

The XML declaration, if it appears at all, must appear on the very first line of the document with no preceding white space. It looks like this.

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

Processing Instructions

Processing instructions are used to pass parameters to an application. These parameters tell the application how to process the XML document. For example, the following processing instruction tells the application that it should transform the XML document using the XSL stylesheet beatles.xsl.

<?xml-stylesheet href="beatles.xsl" type="text/xsl"?>

As shown above, processing instructions begin with and <? end with ?>.

Comments

Comments can appear throughout an XML document. Like in HTML, they begin with <!-- and end with -->.

<!--This is a comment-->

A Document Type Declaration

The Document Type Declaration (or DOCTYPE Declaration) has three roles.

1. It specifies the name of the document element.
2. It may point to an external Document Type Definition (DTD).
3. It may contain an internal DTD.

Empty Elements

Not all elements contain other elements or text. For example, in XHTML, there is an img element that is used to display an image. It does not contain any text or elements within it, so it is called an empty element. In XML, empty elements must be closed, but they do not require a separate close tag

IV. ELEMENTS

Every XML document must have at least one element, called the document element. The document element usually contains other elements, which contain other elements, and so on. Elements are denoted with tags.

V. ATTRIBUTES

XML elements can be further defined with attributes, which appear inside of the element's open tag.

VI. CDATA

Sometimes it is necessary to include sections in an XML document that should not be parsed by the XML parser. These sections might contain content that will confuse the XML parser, perhaps because it contains content that appears to be XML, but is not meant to be interpreted as XML. Such content must be nested in CDATA sections

VII. WHITE SPACE

In XML data, there are only four white space characters.

(1)Tab
(2)Line-feed
(3)Carriage-return
(4)Single space

VIII. XML SYNTAX RULES

XML has relatively straightforward, but very strict, syntax rules. A document that follows these syntax rules is said to be well-formed.

(1)There must be one and only one document Element.
(2)Every open tag must be closed.
(3)If an element is empty, it still must be closed.

Poorly-formed: <tag>
Well-formed: <tag></tag>
Also well-formed: <tag />
(5)Elements must be properly nested.
Poorly-formed: <a>
Well-formed: <a>

(6)Tag and attribute names are case sensitive.
(7)Attribute values must be enclosed in single or double quotes.

IX. JAVA DOM: THE DOCUMENT OBJECT

The DOM Document object represents an XML document. When you parse an XML file using a Java DOM parser, you get back a Document object.

The two most commonly used features of DOM are:

(1)Accessing Child Elements of an Element
(2)Accessing Attributes of an Element

The DOM Document Element

A DOM object contains a lot of different nodes connected in a tree-like structure. At the top is the Document object. The Document object has a single root element, which is returned by calling getDocumentElement().

X. HTTP REQUEST

The following introductory topics will be discussed in this article:

- The life-cycle of an HTTP request & response.
- Anatomy of an HTTP request & response.

HTTP Methods & best practices.

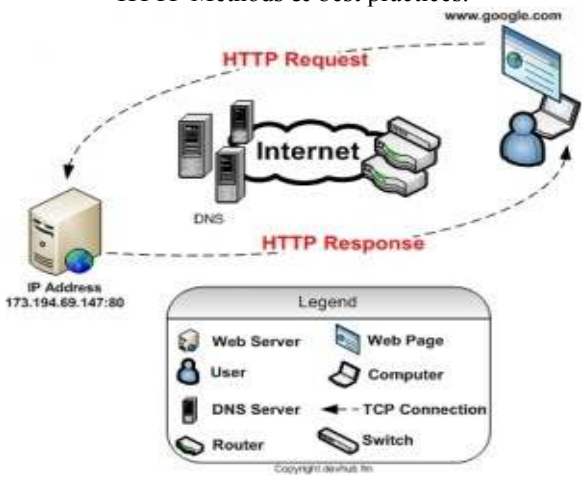

Figure:1

The life-cycle of an HTTP request commonly looks like this:
(1)A user visits the URL of a website.
(2)This creates a request which is routed to a web server via the internet (a network of DNS's, routers and switches) over HTTP (Hypertext Transfer Protocol).
(3)The web server receives the HTTP request and responds to the user with the web page (or content) which was requested.

XI. HTTP RESPONSE CODES

OK 200

The request was fulfilled.

Bad request 400

The request had bad syntax or was inherently impossible to be satisfied.

Internal Error 500

The server encountered an unexpected condition which prevented it from fulfilling the request.

Moved 301

The data requested has been assigned a new URI.

Reading from and Writing to a URLConnection

The URLConnection class contains many methods that let you communicate with the URL over the network. URLConnection is an HTTP-centric class; that is, many of its methods are useful only when you are working with HTTP URLs. However, most URL protocols allow you to read from and write to the connection. This section describes both functions.

Reading from a URLConnection

a URL Connection object and gets an input stream from the connection. The connection is opened implicitly by calling getInputStream. Then a BufferedReader is created in the input stream which reads the data.

```
import java.net.*;
import java.io.*;
public class URLConnectionReader {
   public static void main(String[] args) throws Exception {
      URL oracle = new URL("http://www.oracle.com/");
      URLConnection yc = oracle.openConnection();
      BufferedReader in = new BufferedReader (new
InputStreamReader( yc.getInputStream ()));
      String inputLine;
      while ((inputLine = in.readLine ())! = null)
         System.out.println(inputLine);
      in.close();
   }
}
```

The output from this program is identical to the output from the program that opens a stream directly from the URL. You can use either way to read from a URL. However, reading from a URLConnection instead of reading directly from a URL might be more useful. This is because you can use the URLConnection object for other tasks (like writing to the URL) at the same time.

Writing to a URLConnection

Many HTML pages contain forms — text fields and other GUI objects that let you enter data to send to the server. After you type in the required information and initiate the query by clicking a button, your Web browser writes the data to the URL over the network. At the other end the server receives the data, processes it, and then sends you a response, usually in the form of a new HTML page.

Many of these HTML forms use the HTTP POST METHOD to send data to the server. Thus writing to a URL is often called posting to a URL. The server recognizes the POST request and reads the data sent from the client.
For a Java program to interact with a server-side process it simply must be able to write to a URL, thus providing data to the server. It can do this by following these steps:

1. Create a URL.
2. Retrieve the URLConnection object.
3. Set output capability on the URLConnection.
4. Open a connection to the resource.
5. Get an output stream from the connection.
6. Write to the output stream.
7. Close the output stream.

The anatomy of an HTTP request:

As a web developer, an important area to understand is the method portion of an HTTP request. The method tells the web server what kind of request is being performed on a URI. So if you type in the URL www.google.com/finance (for example). You are requesting the /finance URI. Within the /finance URI the HTTP request has to define an HTTP method.
The method portion of an HTTP request contains the following definition options:

```
Method        = "OPTIONS"
              | "GET"
              | "HEAD"
              | "POST"
              | "PUT"
              | "DELETE"
              | "TRACE"
              | "CONNECT"
              | extension-method
extension-method = token
```

XII. SENDING MAIL THROUGH JAVA

The Java Mail API provides support for sending and receiving electronic mail messages. The API provides a plug-in architecture where vendor's implementation for their own proprietary protocols can be dynamically discovered and used at the run time. Sun provides a reference implementation and its supports the following protocols namely,

- Internet Mail Access Protocol (IMAP)
- Simple Mail Transfer Protocol (SMTP)
- Post Office Protocol 3(POP 3)

The first thing is that a Mail Session has to be established with some properties for sending or receiving a mail. The mandatory property is the server name then, it is optional to provide the port information, and it is needed if it is different from the default port of 25. Then we construct a Message object for the Mail session by populating the information like sender, receiver, subject and text.

Then the message is sent by calling the Transport.send() method.

XIII. COLLECTIONS

Collections (sometimes called containers) are holders that let you store and organize objects in useful ways for efficient access. What will be efficient depends on how you need to use the collection, so collections come in many flavors. Most programming environments provide some collection types, ranging from impoverished up through gargantuan.

In the package java.util you will find interfaces and classes that provide a generic collection framework. This framework gives you a consistent and flexible set of collection interfaces and several useful implementations of these interfaces. You've already been briefly introduced to some of these, such as the interfaces List, Set, Map, and Iterator, and implementations ArrayList and HashMap.

The collection framework is designed to be concise. The principle is to have a core set of valuable collection abstractions and implementations that are broadly useful, rather than an exhaustive set that is complete but conceptually complex and unwieldy.

One way to keep the size down is to represent broad abstractions in the interfaces rather than fine-grained differences. Notions such as immutability and resizability are not represented by different interface types. The core collection interfaces provide methods that allow all common operations, leaving it to specific implementations to refuse to execute particular improper operations by throwing the unchecked java.lang.UnsupportedOperationException.The collections interfaces are:

- Collection<E> The root interface for collections. Provides such methods as add, remove, size, and toArray.
- Queue<E> A collection with an implied ordering in its elements (extends Collection<E>). Every queue has a head element that is the target of specific operations like peek and poll.
- Map<K,V> A mapping from keys to at most one value each. (Map does not extend Collection, although the concepts meaningful to both maps and collections are represented by methods of the same names, and maps can be viewed as collections.)
- SortedMap<K,V> A map whose keys are sorted (extends Map<K,V>).

The interfaces SortedSet and SortedMap guarantee that iteration through the elements is done in sorted order. The java.util package also provides several useful concrete implementations of these interfaces that will suffice for most of your needs. For example:

- TReeSet<E> A SortedSet implemented as a balanced binary tree. Slower to search or modify than a HashSet, but keeps the elements sorted.
- LinkedList<E> A doubly linked List and Queue implementation. Modification is cheap at any size, but random access is slow.
- HashMap<K,V> A hashtable implementation of Map. A very generally useful collection with relatively cheap lookup and insertion times.
- TreeMap<K,V>An implementation of SortedMap as a balanced binary tree to keep its elements ordered by key. Useful for ordered data sets that require moderately quick lookup by key.

Nearly all the implementation classes in java.util are both Cloneable and Serializable. The exceptions are PriorityQueue which is not Cloneable, and WeakHashMap which is neither.

XIV. STEPS FOR MANOWS

1. Create a XML file which contains the URL to be checked and the messages to be sent for some particular status codes. It also contains the email ID to which the notification email is to be sent.
2. The next step is to validate the XML. If the validation is true then goto the next step otherwise repeat this step until it becomes true.
3. Then the XML is parsed using the DOM parser.
4. In this step, the URL request is made which means the particular URL whose status is to be checked makes the request.

5. The status code which is received after sending the request for that particular URL is checked.
6. The email is sent to the email ID mentioned in the XML document for that particular status code.

XV. MAIN RESULTS

Figure:2

Figure: 3

A typical validation requirement for an application written in code or script is to validate an XML document as the document is parsed. The objective is normally to read the document from a disk file or stream, parse it, then process the data as the application requires. As long as the parsing phase doesn't complain as the XML document is read in, the application can take it for granted that the XML document it is working with is valid.

XVI. III. CONCLUSIONS

In today's information age competition, a smoothly operating Web site provides a distinct competitive advantage. On the Internet, the only hard currency is attention. A Web site that fails to deliver its content, either in a timely manner or at all, causes visitors to quickly lose interest, wasting the time and money invested in the web site development. Failure of
Web site quickly sends your potential customers to the "just a click-away" competitor.

Web monitoring is good for business. The Internet as a productivity tool has wide acceptance but recent changes have brought new distractions costing business some of those productivity gains. The Internet can be controlled but needs to be done in a way that allows for employee buy-in, self monitoring and self enforcement to be successful.

REFERENCES

[1] Jian Xu, Manwu Xu, **"A Performance** Monitoring Tool for predicting degradation in Distributed System", WISM 2009, Shanghai, China.

[2] A.van der zee, A.Courbot, T.Nakajima, **"Action-**based performance monitoring of multi-tier **web application"**, CSE 2009, Vancouver, Canada.

[3] Feng Liu, Zhenming Lei, Hui Miao, **"Web** performance analysis on real network", ICECC 2011, Ningbo, China.

[4] Thomas M. Chen, Lucia Hu, **"Internet** Performance Monitoring", IEEE, VOL. 90, NO. 9, 2002.

[5] K. Thompson, G. Miller, and R. Wilder, "**Wide-** area internet traffic patterns and **characteristics**," *IEEE Network*, vol. 11, pp. 10–23, Nov.1997

[6] K. Claffy and T. Monk, "**What's next for internet** data analysis? Status and challenges facing **the community**," *Proc. IEEE*, vol. 85,pp. 1563–1571, Oct. 1997.

[7] V. Paxson *et al.*, "An architecture for large-scale internet measurement," *IEEE Commun. Mag.*, vol. 36, pp. 48–54, Aug. 1998

[8] J.-S. Park, J.-Y. Lee, and S.-B. Lee, "traffic measurement and analysis in a high speed network environment: Workload and flow characteristics," *J. Commun. Networks*, vol. 2, pp. 287–296, Sept. 2000.

[9] B. Braden *et al.*, "**Resource reservation** protocol (RSVP) – Version 1 functional **specification**," IETF RFC 2205, 1997.

[10] M. Crovella and A. Bestavros, "**Self-similarity** in World Wide Web traffic: Evidence and possible causes," *IEEE/ACM Trans.Networking*, vol. 5, pp. 835–846, Dec. 1997.

[11] A. Downey, "Using pathchar to estimate internet link characteristics," *Comp. Commun.Rev.*, vol. 29, pp. 241–250, Oct. 1999.

[12] **Internet Traffic Report**. [Online]. http://www.Internet-TrafficReport.com.

[13] S. Kalidindi and M. Zekauskas, "**Surveyor: An** infrastructure for internet performance **measurements**," in INET'99, June 1999.

[14] T. Chen *et al.*, "Monitoring and control of ATM networks using special cells," *IEEE Network*, vol. 10, pp. 28–38, Sept. 1996.

[15] "B-ISDN operation and maintenance principles and functions," ITU-T Rec. I.610, Geneva, Switzerland, 1993.

AUTHORS

First Author – Pratyush Poddar, B.Tech (CS&E), Amity University, Noida, pratyushpoddar123@gmail.com
Second Author – Praveen Kumar, Assistant Professor, Amity University, Noida, pkumar3@amity.edu