Design and Development of Website Validator using XHTML 1.0 Strict Standard

Ibnu Gunawan
Informatics Department
Petra Christian University
Surabaya, Indonesia
Ibnu@petra.ac.id

Arlinah Imam Rahardjo
Informatics Department
Petra Christian University
Surabaya, Indonesia
Arlinah@petra.ac.id

Yohanes Edwin
Informatics Department
Petra Christian University
Surabaya, Indonesia
M26407106@petra.ac.id

Based on research done by w3.org[1] there are five reasons why we must validate our web site : Validation as a debugging tool, Validation as a future-proof quality check, Validation eases maintenance, Validation helps teach good practices, Validation is a sign of professionalism. Therefore, we need an application that can help web designer create a good website with a valid HTML element and CSS.

This paper will show how to design and develop that application. Validation system process begins with lexical analysis, parsing, and then analysis of the tokens. Tokens being analyzed are elements and attributes. These tokens are checked for compliance with the Document Type Definition (DTD) Standard. In addition to the tokens, this application checks web page structures too. This application can also perform CSS validation in a web page. Moreover, the user can directly correct an invalid web page, thus shortens correction time. This application was created using Microsoft Visual Basic .NET 2008 as a programming framework and Microsoft SQL Server 2008 Express as a database.

Based on application testing, it can be concluded that the application can find errors either to validate HTML code or CSS validation. This application can also improve the web page complying with DTD XHTML 1.0 Strict Standard.

Keywords-component; Markup Validation, HTML, CSS, DTD, XHTML 1.0 Strict, Website Validator.

I. INTRODUCTION

Based on question that asked by w3.org to the web community in 2009 [1] there are some reason how we must validate our website, like:

1. Validation as a debugging tool

   While contemporary Web browsers do an increasingly good job of parsing even the worst HTML "tag soup", some errors are not always caught gracefully. Very often, different software on different platforms will not handle errors in a similar fashion, making it extremely difficult to apply style or layout consistently.

   Using standard, interoperable markup and style sheets, on the other hand, offers a much greater chance of having one's page handled consistently across platforms and user-agents. Indeed, most developers creating rich Web applications know that reliable scripting needs the document to be parsed by User-Agents without any unexpected error, and will make sure that their markup and CSS is validated before creating a rich interactive layer.

   When surveyed, a large majority of Web professionals will state that validation errors is the first thing they will check whenever they run into a Web styling or scripting bug.

2. Validation as a future-proof quality check

   Checking that a page "displays fine" in several contemporary browsers may be a reasonable insurance that the page will "work" today, but it does not guarantee that it will work tomorrow.

   In the past, many authors who relied on the quirks of Netscape 1.1 suddenly found their pages appeared totally blank in Netscape 2.0. Whilst Internet Explorer initially set out to be bug-compatible with Netscape, it too has moved towards standards compliance in later releases.

   Validation is one of the simplest ways to check whether a page is built in accordance with Web standards, and provides one of the most reliable guarantees that future Web platforms will handle it as designed.

3. Validation eases maintenance

   It is reasonable to consider that standards such as HTML and CSS are a form of "coding style" which is globally agreed upon. Creating Web pages or applications according to a widely accepted coding style makes them easier to maintain, even if the maintenance and evolution is performed by someone else.

4. Validation helps teach good practices

   Many professionals have been authoring the Web with HTML and CSS for years and know these technologies by heart. Beginners and students, on the other hands, will find automated checking tools invaluable in spotting mistakes. Some teachers also stress that automated validation tests are a good introduction to broader, more complex quality concepts such as accessibility.

5. Validation is a sign of professionalism

   As of today, there is little or no certification for Web professionals, and only few universities teach Web technologies, leaving most Web-smiths to learn by themselves, with varied success. Seasoned, able professionals will take pride in creating Web content using semantic and well-formed markup, separation of style and content, etc. Validation can then be used as a quick check to determine whether the code is the clean work of a seasoned HTML author, or quickly hacked-together tag soup.

Because the importance of this validation process, we build a web validator that not only can do the validation but can fix the source code too.

II. BACKGROUND

In order to validate website, our validator used xhtml 1.0 strict standard and there are 3 processes involved, begin with lexical analysis, continued with parsing, and then finished by token analysis. The following detailed explanation of what is it xhtml 1.0 strict standard, css, lexical analysis and parsing.

A. XHTML 1.0 Strict

December 1998 saw the publication of a W3C Working Draft entitled *Reformulating HTML in XML*. This introduced Voyager, the codename for a new markup language based on HTML 4, but adhering to the stricter syntax rules of XML. By February 1999 the name of the specification had changed to *XHTML 1.0: The Extensible HyperText Markup Language*, and in January 2000 it was officially adopted as a W3C Recommendation[2]. There are three formal DTDs for XHTML 1.0, corresponding to the three different versions of HTML 4.01:

1. XHTML 1.0 Strict
   The XML equivalent to strict HTML 4.01, and includes elements and attributes that have not been marked deprecated in the HTML 4.01 specification. As of May 25 2011, XHTML 1.0 Strict is the document type used for the homepage of the website of the World Wide Web Consortium.
2. XHTML 1.0 Transitional
   The XML equivalent of HTML 4.01 Transitional, and includes the presentational elements (such as `center`, `font` and `strike`) excluded from the strict version.
3. XHTML 1.0 Frameset
   The XML equivalent of HTML 4.01 Frameset, and allows for the definition of frameset documents—a common Web feature in the late 1990s.

B. CSS (Cascading Style Sheets)

Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation semantics (the look and formatting) of a document written in a markup language. It's most common application is to style web pages written in HTML and XHTML, but the language can also be applied to any kind of XML document, including plain XML, SVG and XUL.

CSS is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation, including elements such as the layout, colors, and fonts[3]. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content (such as by allowing for table less web design).

CSS can also allow the same markup page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices. While the author of a document typically links that document to a CSS style sheet, readers can use a different style sheet, perhaps one on their own computer, to override the one the author has specified.

C. Lexical Analysis

Lexical analysis or *scanning* is the process where the stream of characters making up the source program is read from left-to-right and grouped into tokens. Tokens are sequences of characters with a collective meaning. There are usually only a small number of tokens for a programming language: constants (integer, double, char, string, etc.), operators (arithmetic, relational, logical), punctuation, and reserved words [4].

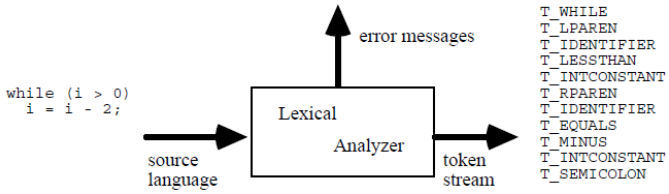

Figure1. Lexical Analyzer diagrams

The lexical analyzer takes a source program as input, and produces a stream of tokens as output. The lexical analyzer might recognize particular instances of tokens such as:

3 or **255** for an integer constant token
"Fred" or **"Wilma"** for a string constant token
numTickets or **queue** for a variable token

Such specific instances are called *lexemes*. A lexeme is the actual character sequence forming a token; the token is the general class that a lexeme belongs to. Some tokens have exactly one lexeme (e.g., the > character); for others, there are many lexemes (e.g., integer constants).

D. Parsing

Parsing is a process that analyzes a token or a few sequences to determine the grammatical structure of a programming language [5]. Parsing is a process of converting a text into a tree. This tree will be processed further.

Parsing the first stage of the process is lexical analysis, where an input is converted into meaningful tokens. This token is then stored in a tree. The next stage is the stage syntactic analysis, namely the examination stage of the token to the standard expression. The last stage is the semantic parsing; here the code will be generated into a result.

III. SYSTEM DESIGN

Our Web site validator will follow 3 main sequence as a system. Begin with input (tahap sistem input), continued by processing (tahap pemrosesan validasi) and last one is output (tahap sistem output). Figure 2 show the system flowchart as a whole.

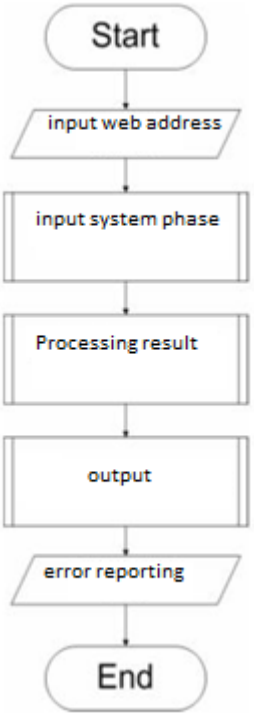

Figure 2. Whole web site validator system

To condense the explanation, we will explain all this process using a flowchart begins with input system.

A. System input flowchart

Input system in this application comes from the Internet URL appears in an input box. Applications will first check the validity of the URL. If the input is really a URL then the application will take the source code from the URLs that are inputted and wrote it in a Text Box. Then the user chooses to perform validation of HTML or CSS validation. After that, the text in the Text Box was put into a string variable that will be processed at a later stage. The flowchart shown in figure 3.

B. System Process flowchart

Figure 4 show main process flowchart. Input is entered in this system is the text that comes from a Text Box. While the output of this system is the result of checking against text that has been validated. Text is entered in this system will be read from top to finish. Any errors found are stored in a variable which will be output from these systems.

If we choose XHTML validation (flowchart is on Figure 6.) then this sequence is executing by computer:

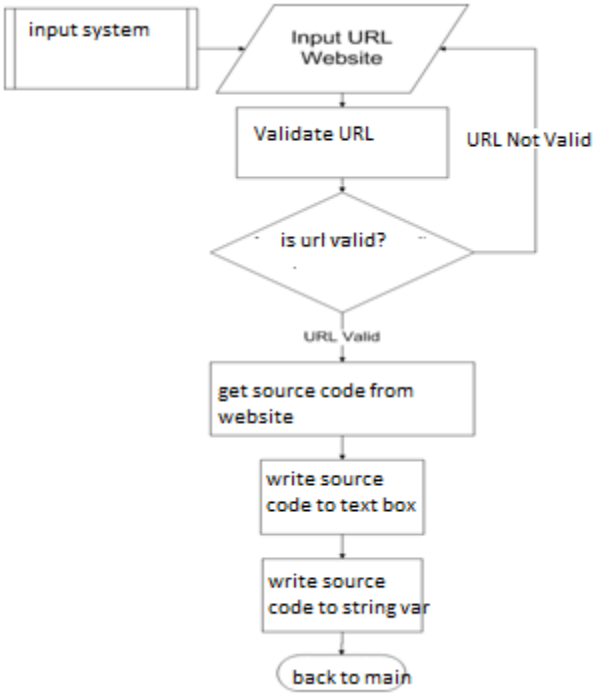

Figure 3. Input system flowchart

- Lexical Analysis Process XHTML
- Token Parsing
- Token Examination

Whereas if the selected CSS validation (flowchart shown on Figure 5), then the processes that occur in this system will be:

- CSS Validation using W3C CSS Validation Web Service
- Making Process Validation results in the form of XML files
- The process of reading an XML file results

Figure 4. Main system process flowchart

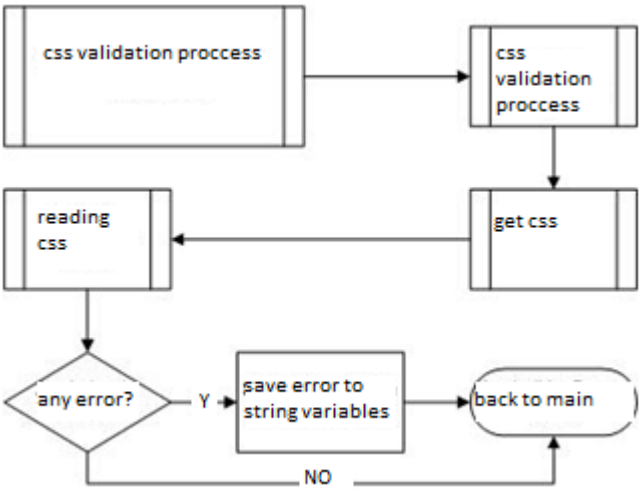

Figure 5. CSS process flowchart

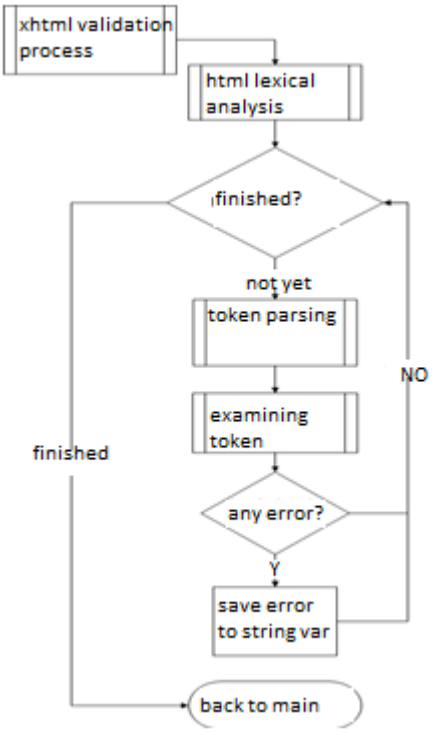

Figure 6. XHTML validation process flowchart.

After we show the main process flowchart followed by 2 sub main flowchart i.e. CSS validation and XHTML validation, It's good to show the detail of css validation and xhtml validation. We begin by xhtml validation, xhtml validation consist of 3 processes:

- Lexical analysis

  The first step taken in the Lexical Analysis is the process of reading the text in the Text Box that has been opened. This process will take tokens that will be used for XHTML validation system. Text is stored into a string variable. This string variable will flow in through the Lexical Analysis process until the entire text is read out. When the process find the token, then the results will be checked its validity in the examination process token validation

  Lexical Analysis of this process begins with finding the DTD used by the XHTML file. DTD that is used should be a DTD XHTML 1.0 Strict, if not a DTD XHTML 1.0 Strict, and then the inspection process was not continued. If the HTML file being examined does have a DTD XHTML 1.0 Strict, then the process Lexical Analysis will seek the position of the characters '<', where the characters '<' is the opening character of the tag that is on an XHTML file.

  After finding the characters '<', then the process Lexical Analysis will proceed by finding the position of the character '>'. The character '>' is a cover of a character element in the XHTML file. If the process did not find the character '>' after '<' then the system will log the error and continue the process of Lexical Analysis. At the time of switching line, the system also will record the current line is checked, so when the process found an error it can be seen also the fault line.

- Token Parsing

  This parsing process is a second process that is used in determining the validity of an XHTML file. This process is done after the program to get tokens from the Lexical Analysis.

  Inputs for this process are the tags that Lexical Analysis obtained from the process before. The tags are available will be included in a stack. Tags that are in the stack will be inspected at the next process. Tag storage is limited only to HTML elements only, while for comments or strings that are not preceded by the character '<' will not be examined. This process flowchart show on figure 7

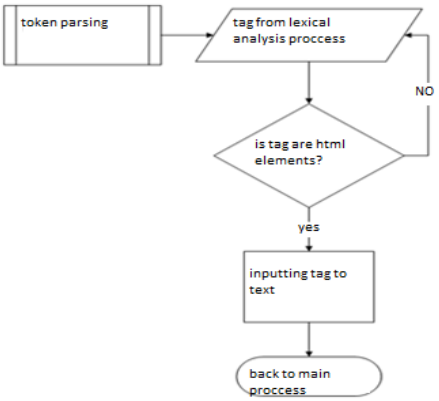

Figure 7. Token Parsing Process flowchart

- Token inspection

  This inspection process is the last of the three programs to validate the XHTML file. This process is done right after the process of parsing and generating elements have been incorporated into the stack.

This inspection process through several stages. The first thing to do are carried out basic checks, like checking a large / small letters, checking what elements are allowed to be used in DTD XHTML 1.0 Strict, and also cover for each element that has been opened. After the basic checks, the system will do further investigation, further investigation is done by examining the sequence element to conform to the rules of a correct XHTML, for example, only have one html element, and also whether, after the html element is found then the next element is checked is the head element or not.

If, this process produce an errors then the error message will be stored in a variable. An error message is displayed when the program is completed through the checking process.

Now the time for detail of three CSS validation process, and because all of the three process is simple enough so we will show all three CSS validation sub main process in flowchart on Figure 8, and 9.

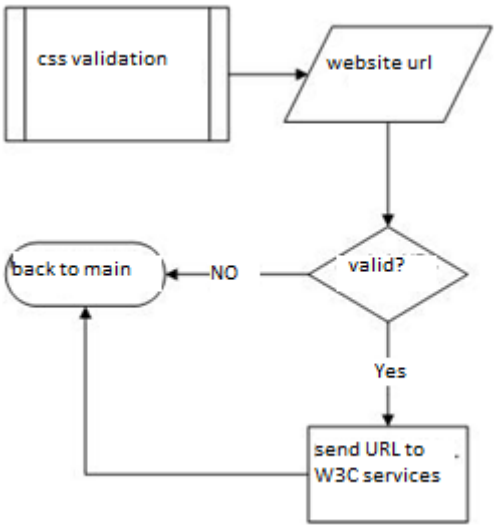

Figure 8. CSS Validation

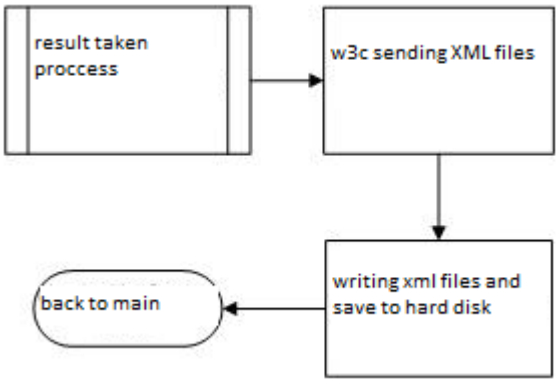

Figure 9. Taken XML standard

C. System Output

The system output is the last system that performed on this program. Once the file is XHTML / CSS has been checked in the previous system, the result of an error message will be displayed on this system. This message will be displayed in a Text Box so the user can see immediately that there are errors in the XHTML file are validated.

Error messages are displayed on the Text Box contains errors and line where the error is located. In addition, for the inspection of XHTML, if the remaining elements in the stack, then the element will be considered a fault as well because it has not closed / unopened. If the previous inspection there was no error message, then the system will write the XHTML file that is checked is valid according to DTD XHTML 1.0 Strict.

IV. System Implementation and Testing

We implement the system using VB programming language. NET applications that have been integrated with Microsoft Visual Studio 2008. This application is made using VB programming language. NET because the source code in VB. NET can be better understood by the layman. In addition, the Microsoft Visual Studio 2008 is also equipped with facilities to design GUI (Graphical User Interface) which is easy to use to use and also the facility to use Microsoft SQL Express database.

A. Libraries and Headers used

In making the application of this Thesis is used namespace (a collection of classes that have been rolled into one) that has been provided by Microsoft Visual Studio 2008. Namespace - namespace used for the manufacture of these applications include:

- `System.IO`, used to read the source code of a website, writes the source code of the website into a text file, and also provide reports final results of the validity of a website.
- `System.Text`, used to store the entire HTML code that has been obtained from the internet which is then used for the validation process.
- `System.Text.RegularExpressions`, used to search a text within the HTML code making it easier for validation.
- `System.Net`, used to make the connection to the internet and also make requests to a website.
- `System.Xml`, used to create, write, and read an XML file.
- `System.Data.SqlClient`, is used to connect to Microsoft SQL Server Express and also the use of Data Adapter / Command Builder of SQL Server.

B. System Testing

Application validation testing was conducted to determine the validity of the application when compared to similar applications that already exist. Tests to be performed include:

- Validation testing websites with XHTML 1.0 Strict DTD.
- CSS validation testing on the website.
- Testing results of repair of the wrong web page.
- The first test, validation on web pages using similar applications.
- The second test, validation on web pages using similar applications.
- The third test, checking the results of improvements on the website page by using the application validator of the W3C (World Wide Consortium.

We have been captured some of the testing screen and shown in figure 10 to 12.

Figure 10. XHTML valid testing

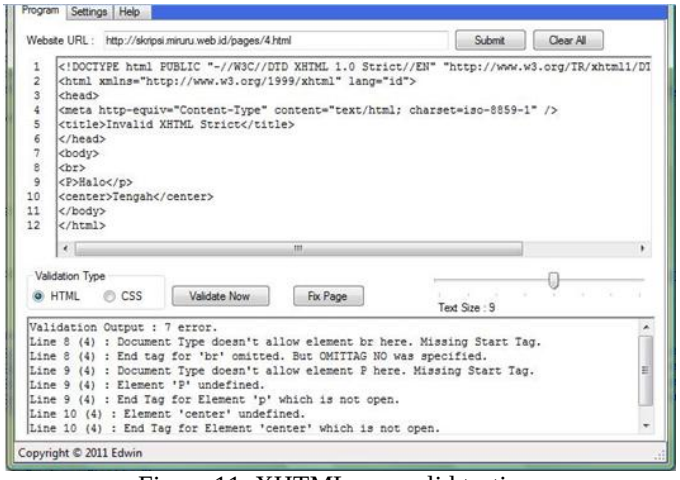

Figure 11. XHTML non valid testing

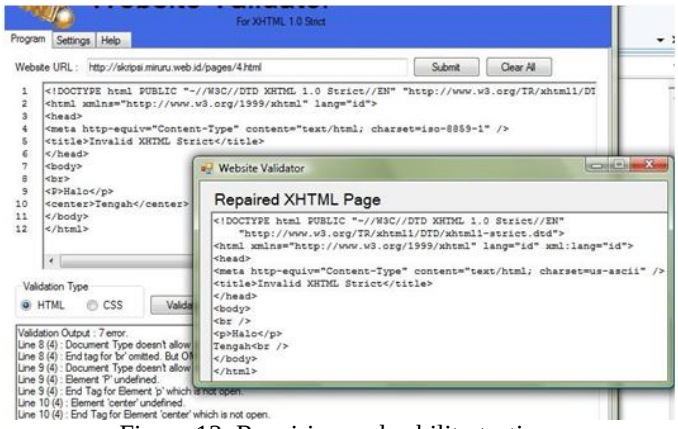

Figure 12. Repairing code ability testing

And table 1 show the accuracy of repairing ability from our web validator application (aplikasi skripsi) compare to w3c validator itself and an open source application on www.mousehuntgame.com for a case study

TABLE 1. ACCURACY COMPARISON

	Aplikasi Skripsi	*W3C Validator*	*Open Source*
Before	63 *Error*	74 *Error*	59 *Error*
after	5 *Error*	7 *Error*	3 *Error*

V. CONCLUSIONS

From the Design and Development of Website Validator using XHTML 1.0 Strict Standard we can obtain several conclusions as follows:

1. The more complete the database owned by an application of the more complete validation of all errors that can be recorded.
2. According to table 1, For detecting error, Our Application is more accurate than Open Source validator . For fixing error, Our Application is more better than W3C Validator.

REFERENCES

[1] Théreaux Olivier, "Valid sites work better(?)," w3c blog, January 2009.

[2] XHTML 1.0: The Extensible HyperText Markup Language, W3C Recommendation 26 January 2000". World Wide Web Consortium. 2000-01-26. Retrieved 2008-07-19.

[3] What is CSS?". World Wide Web Consortium. Retrieved December 2010.

[4] Johnson Maggie and Zelenski Julie, "lexical analysis handout for cs143," Stanford University, Summer 2008.

[5] Yulia, "Perancangan dan pembuatan sistem validasi XHTML 1.0," Paper presented at Konferensi Nasional Sistem dan Informatika 2008, Bali, Indonesia.