HTML VALIDATION THROUGH EXTENDED VALIDATION SCHEMA

Radoslav Radev

Abstract: *The paper presents extensible software architecture and a prototype and an implementation of a highly configurable system for HTML validation. It is based on validation rules defined in an XML document called "extended validation schema". It serves as an extended validation schema beside the official HTML specification, because the browsers' and other web clients' differences in HTML visualization makes the HTML specification insufficient and it is perfectly possible an HTML document to be syntax valid and yet not well visualized in some browser or mail-client. The extended validation schema allows definition of custom and specific validation rules in three levels - document rules, element (or tag) rules and attributes rules. The correctness of the validation schema is checked via a predefined XSD schema. The paper defines a prototype of a validation engine that consists of HTML parser, HTML validator, Storage module and Statistics module. The HTML parser parses the HTML file and breaks it into corresponding elements. The HTML validator applies the custom validations defined in the extended validation schema for every single element and attribute along with document-level validations, and also automatically corrects the errors wherever possible. The Storage module saves the validation results to a persistent storage. They can be considered for unit tests and used by the Statistics module to create additional statistics, analyses, quality assurance and bug tracking. A comparison is made with other HTML validation services and solutions. The results of an implementation of the prototype system in a software company are also presented.*

Keywords: *HTML validation, XML schema, quality assurance, unit tests, bugs tracking.*

ACM Classification Keywords: *D.m Software – Miscellaneous.*

Introduction

The final output of a typical real world web-application today is an HTML/CSS/Java Script code. Web browsers interpret this code and visualize it to the final user. Because of the different manner they do it, in practice it is difficult to write a HTML/CSS/Java Script code that will be visualized in a correct way. If the code is meant to be visualized not only in a browser but by an e-mail or a mobile client also, this means additional limitations and troubles. Actually the perfect HTML/CSS/Java Script code must match the intersection of the supported functionality of all the clients that are expected to visualize it.

The experienced front-end developers know good patterns for avoiding potential problems. They are not documented in a single global reference but exist only as a personal experience and knowledge. The beginners usually learn them with practice, causing a lot of errors, effort, and time. So it is good and even required for a front-end developer or team to have automatic validation functionality.

There are many HTML validation services, for example W3C Markup Validation Service [W3C Service]. They validate the HTML code against the rules of the HTML specification [HTML Specification]. They identify syntax errors in the HTML code [Chen, 2005]. But usually there are many other "unofficial" rules, defined only as a common agreement between the members of the development team. Practically the HTML specification is only the solid bases that must be followed. On top of it there are many other specific rules depending on the software project, client, task, team organization and other factors. So it is perfectly possible an HTML file to be valid according to HTML specification and still not visualized as intended by the browsers and other clients. These specific factors in validation make it impossible to create a universal validator.

The approach suggested in this paper is to propose a configurable validation schema defined as an XML file. It is subsequently executed against the HTML code. In this way every developer or team can easily define its own specific validation rules and track their satisfaction for a period in time. The most common mistakes can be identified and corrected, and for example a specific training course for developers can be organized according the results.

Objectives

The first objective of this paper is to create a validation schema that allows developers to define specific validation rules beside the official HTML specification. For this reason three types of rules are defined:

1. Document-level rules:
   - The document must have a specified encoding.
   - The document must have a specified document type.
   - The document must contain a specific JavaScript code.
   - All special symbols in the document must be encoded.
   - The document must not contain any comments.
2. Element (tag) level rules:
   - The document must contain a meta tag with a specified content.
   - The document must not contain a meta tag with a specified content.
   - A specified tag must not be used in the HTML document.
   - A specified tag must not be empty (must have at least one child).
   - A specified start tag and its corresponding end tag must be on the same line in the code.
   - A specified tag to be treated as correct even if it does not exists in the HTML specification.
3. Attribute level rules:
   - A specified tag must contain a specified attribute.
   - A specified tag must contain a specified attribute and its value must not be empty.
   - A specified tag must contain a specified attribute and its value must match a specified value.
   - A specified tag must contain a specified attribute and it must not contain spaces around its value.
   - A specified tag must contain a specified attribute and it must not contain spaces around its equal sign.
   - A specified tag must contain a specified attribute and it must not contain spaces in its value.
   - A specified tag must contain a specified attribute and its value must match a specified regular expression.
   - A specified tag must not contain a specified attribute.
   - A specified tag must not contain any attributes except the specified ones.

The second objective of this paper is to create a prototype of a system that:

- Receives a validation schema as an XML file and the HTML document to be validated.
- Performs the validations defined in the schema against the document.
- Performs automatic correction of the errors wherever possible.

- Visualize the validation results.
- Saves the validation results to a persistent storage.
- Creates and visualizes statistics and analysis based on the validation results in the persistent storage.

Validation Schema

The validation schema proposed in this paper is an XML file. Here is a real-world example of a validation schema with full functionality:

```
<?xml version="1.0" encoding="utf-8" ?>
<ValidationManager xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" DocType="HTML 4.01 Transitional" Encoding="UTF-8">
 <Tags>
  <ValidationTag Tag="a">
   <Attributes>
    <ValidationAttribute Name="target" IsRequired="true" Value="_blank" />
    <ValidationAttribute Name="title" IsRequired="true" AllowSpaceAroundEqualSign="false"
AllowSpaceAroundValue="false" />
    <ValidationAttribute Name="rilt" IsRequired="true" AllowEmpty="false" AllowSpaceInValue="false"
AllowSpaceAroundEqualSign="false" AllowSpaceAroundValue="false" />
    <ValidationAttribute Name="href" AllowSpaceAroundValue="false" AllowSpaceInValue="false">
     <Expressions>
      <ValidationExpression Expression="^http:\/\/" Message="Link href does not start with http://." />
     </Expressions>
    </ValidationAttribute>
   </Attributes>
  </ValidationTag>
  <ValidationTag Tag="img">
   <Attributes>
    <ValidationAttribute Name="alt" IsRequired="true" AllowSpaceAroundEqualSign="false"
AllowSpaceAroundValue="false" />
    <ValidationAttribute Name="display" IsRequired="true" Value="block" IsStyle="true" />
   </Attributes>
  </ValidationTag>
  <ValidationTag Tag="p" IsAllowed="false" />
  <ValidationTag Tag="table">
   <Attributes>
    <ValidationAttribute Name="width" AllowEmpty="false" />
   </Attributes>
  </ValidationTag>
  <ValidationTag Tag="td">
   <Attributes>
    <ValidationAttribute Name="width" AllowEmpty="false" />
   </Attributes>
  </ValidationTag>
  <ValidationTag Tag="th">
   <Attributes>
    <ValidationAttribute Name="width" AllowEmpty="false" />
   </Attributes>
  </ValidationTag>
  <ValidationTag Tag="tr" AllowEmpty="false" />
 </Tags>
 <MetaTags>
  <ValidationMetaTag IsRequired="true">
   <Attributes>
    <ValidationAttribute Name="http-equiv" Value="Content-Type" />
     <ValidationAttribute Name="content">
      <Expressions>
       <ValidationExpression                Expression="^text/html;(\s*)charset=utf-8;(\s*)width=device-width,(\s*)initial-
scale=0.75$" />
      </Expressions>
     </ValidationAttribute>
    <ValidationAttribute Name="name" Value="viewport" />
   </Attributes>
  </ValidationMetaTag>
 </MetaTags>
 <SpecialChars ErrorMessage="Not encoded special symbol.">¡ ¢ £ ¤ ¥ ¦ § ¨ © ª « ¬ - ® ¯ ° ± ² ³ ´ µ ¶ • ¸ ¹ º » ¼
½ ¾ ¿ À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï Ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð ñ ò ó ô
õ ö ÷ ø ù ú û ü ý þ ÿ Œ œ Š š Ÿ ƒ ˆ ˜ Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο Π Ρ Σ Τ Υ Φ Χ Ψ Ω α β γ δ ε ζ η θ ι κ λ μ ν ξ
ο π ρ ς σ τ υ φ χ ψ ω ϑ ϒ ϖ       – — ‘ ’ ‚ “ ” „ † ‡ • … ‰ ′  ″  ‹ › ‾ ⁄ € ℑ ℘ ℞ ™ ℵ ← ↑ → ↓ ↔ ↵ ⇐ ⇑
⇒ ⇓ ⇔ ∀ ∂ ∃ ∅ ∇ ∈ ∉ ∋ ∏ ∑ − ∗ √ ∝ ∞ ∠ ∧ ∨ ∩ ∪ ∫ ∴ ∼ ≅ ≈ ≠ ≡ ≤ ≥ ⊂ ⊃ ⊄ ⊆ ⊇
⊕ ⊗ ⊥ ⋅ ⌈ ⌉ ⌊ ⌋ ◊ ♠ ♣ ♥ ♦ </SpecialChars>
</ValidationManager>
```

Here is the XSD schema that any validation schema must match:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="ValidationManager" xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="ValidationManager">
  <xs:complexType>
   <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Tags">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="ValidationTag" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="Attributes" minOccurs="0" maxOccurs="1">
           <xs:complexType>
            <xs:sequence>
             <xs:element name="ValidationAttribute" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
               <xs:sequence>
                <xs:element name="Expressions" minOccurs="0" maxOccurs="1">
                 <xs:complexType>
                  <xs:sequence>
                   <xs:element name="ValidationExpression" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                     <xs:attribute name="Expression" type="xs:string" use="required" />
                     <xs:attribute name="Message" type="xs:string" use="required" />
                    </xs:complexType>
                   </xs:element>
                  </xs:sequence>
                 </xs:complexType>
                </xs:element>
               </xs:sequence>
               <xs:attribute name="Name" type="xs:string" use="required" />
               <xs:attribute name="IsStyle" type="xs:boolean" />
               <xs:attribute name="IsRequired" type="xs:boolean" />
               <xs:attribute name="Value" type="xs:string" />
               <xs:attribute name="AllowEmpty" type="xs:boolean" />
               <xs:attribute name="IsAllowed" type="xs:boolean" />
               <xs:attribute name="AllowSpaceAroundEqualSign" type="xs:boolean" />
               <xs:attribute name="AllowSpaceAroundValue" type="xs:boolean" />
               <xs:attribute name="AllowSpaceInValue" type="xs:boolean" />
              </xs:complexType>
             </xs:element>
            </xs:sequence>
           </xs:complexType>
          </xs:element>
         </xs:sequence>
         <xs:attribute name="Tag" type="xs:string" use="required" />
         <xs:attribute name="IsOnTheSameLine" type="xs:boolean" />
         <xs:attribute name="AllowEmpty" type="xs:boolean" />
         <xs:attribute name="IsAllowed" type="xs:boolean" />
         <xs:attribute name="AllowOtherStyleAttributes" type="xs:boolean" />
        </xs:complexType>
       </xs:element>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="SpecialChars" nillable="true">
     <xs:complexType>
      <xs:simpleContent>
       <xs:extension base="xs:string">
        <xs:attribute name="ErrorMessage" type="xs:string" use="required" />
       </xs:extension>
      </xs:simpleContent>
     </xs:complexType>
    </xs:element>
    <xs:element name="Comments">
     <xs:complexType>
      <xs:attribute name="AllowComments" type="xs:boolean" />
      <xs:attribute name="ErrorMessage" type="xs:string" use="required" />
     </xs:complexType>
    </xs:element>
    <xs:element name="AllowedTags">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="Tag" type="xs:string" minOccurs="0" />
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="MetaTags">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="ValidationMetaTag" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="Attributes" minOccurs="1" maxOccurs="1">
           <xs:complexType>
            <xs:sequence>
             <xs:element name="ValidationAttribute" minOccurs="1" maxOccurs="unbounded">
              <xs:complexType>
               <xs:sequence>
                <xs:element name="Expressions" minOccurs="0" maxOccurs="1">
                 <xs:complexType>
                  <xs:sequence>
                   <xs:element name="ValidationExpression" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                     <xs:attribute name="Expression" type="xs:string" use="required" />
                    </xs:complexType>
                   </xs:element>
                  </xs:sequence>
                 </xs:complexType>
                </xs:element>
               </xs:sequence>
               <xs:attribute name="Name" type="xs:string" use="required" />
               <xs:attribute name="Value" type="xs:string" />
              </xs:complexType>
             </xs:element>
            </xs:sequence>
           </xs:complexType>
          </xs:element>
         </xs:sequence>
         <xs:attribute name="IsAllowed" type="xs:boolean" />
         <xs:attribute name="IsRequired" type="xs:boolean" />
        </xs:complexType>
```

```
      </xs:element>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:choice>
  <xs:attribute name="DocType" type="xs:string" />
  <xs:attribute name="Encoding" type="xs:string" />
 </xs:complexType>
</xs:element>
</xs:schema>
```

Validation Engine

The program performing the validation of the HTML document against the validation schema is called validation engine. It consists of the following modules:

1. User interface to receive the HTML document to be validated with following standard possibilities:

1.1. Enter the document as text.

1.2. Upload the document as a file.

1.3. Read the document from an URI.

2. HTML parser:

2.1. Parses the HTML document.

2.2. Breaks it into corresponding parts.

3. HTML validator:

3.1. Performs the validation against the official HTML specification.

3.2. Loads the specified validation schema from an XML file.

3.2. Validates the validation schema against the XSD schema that it must match.

3.3. Iterates through the parts of the HTML document and applies the validation rules defined for any element and attribute.

3.4. If an automatic correction of some errors is possible, it is performed accordingly.

3. Storage module – it saves the errors found in a persistent storage, for example a relational database and in a log file. Automatic corrected errors are also saved to the persistent storage.

4. User interface to display the errors found, each of them consisting of:

4.1. User-friendly error message.

4.2. Line and column in the HTML code.

5. Statistics module – it displays tables, diagrams and charts, for example:

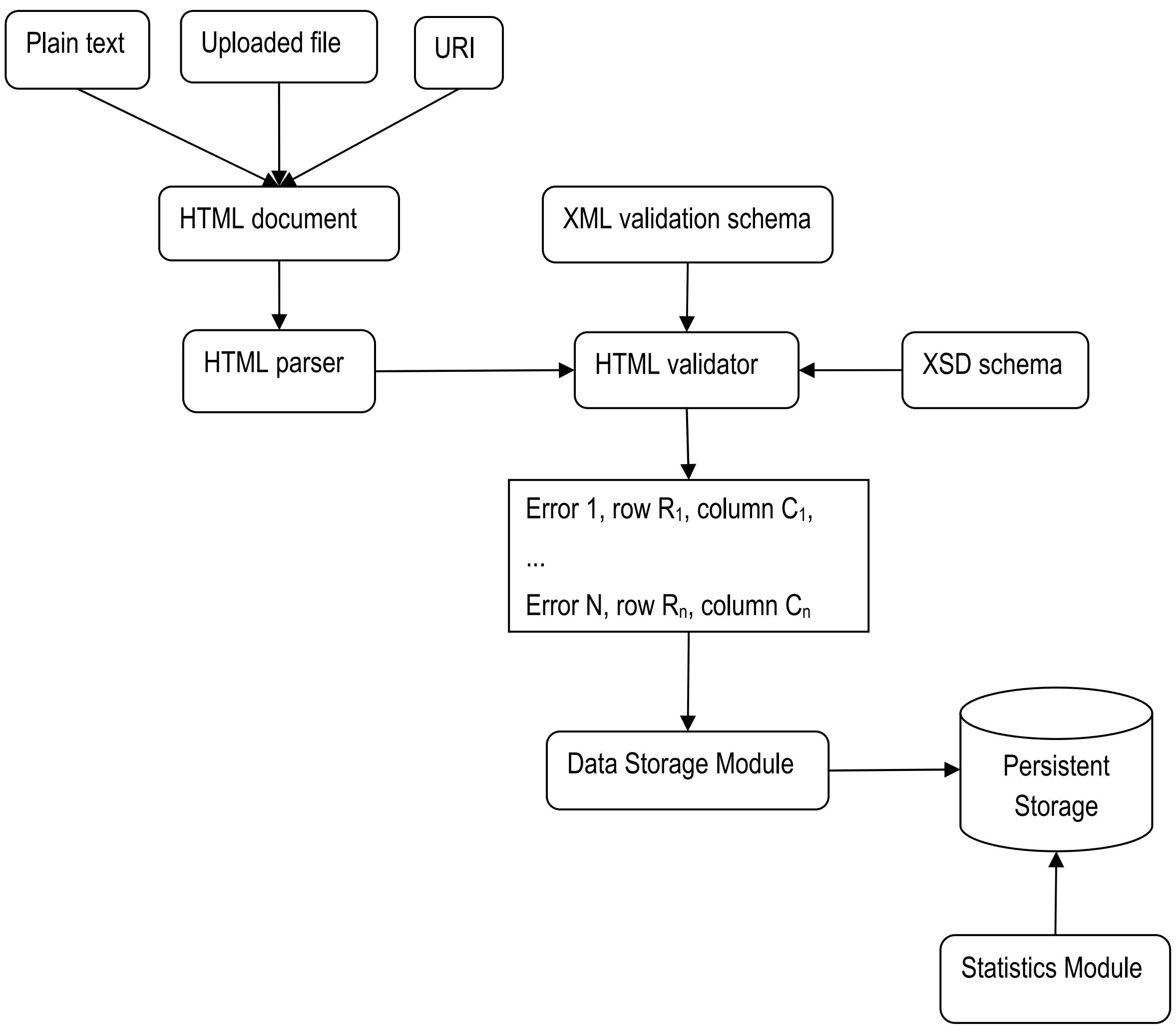
5.1. Most common validation errors (most common incorrect elements and attributes).

5.2. Average number of validations performed for a HTML document.

5.3. The trend of a specific error for a period of time – i.e. bug tracking.

All these statistics may be created for a single developer or for the team as a whole. This allows measuring of creativity and advance of the front-end developers for a period of time.

The following diagram illustrates the validation process:



Comparison with other HTML validation services

There are many HTML validation services and tools, for example:

- W3C Markup Validation Service [W3C Service]
- WDG HTML Validator [WDG]
- CSE HTML Validator [CSE]
- Validome Validation Service [Validome]

They provide syntax checking of the HTML code but do not allow custom validation rules to be added. The solution proposed in this paper seems unique in this area, although may be many companies have implemented some specific internal and not documented solutions according their needs. This paper tries to generalize the extended HTML validation and to serve as a basis for future solutions and implementations.

Conclusion

The proposed validation schema and validation process was integrated in the software company “Stanga” Ltd. in 2010 with the following technologies:

- Programming language: C# 4.0.
- User interface: ASP .NET 4.0.
- Integrated Development Environment: Visual Studio 2010.
- Database: Microsoft SQL Server 2008.

In practice it turned out to be a very useful solution with a perspective for extension in the following directions:

- More validation rules to be added.
- To create an application programming interface (API) to allow third party integration and further customization of the validation process.
- Performance measurements of the validation process and additional logging features.

A test version of the system can be found at: http://qatool.dev.provisionsofia.com/

Bibliography

[W3C Service] W3C Markup Validation Service, http://validator.w3.org/

[HTML Specification] HTML 4.01 Specification, http://www.w3.org/TR/html4/

[Chen, 2005] Chen, Sh., Hong, D., Shen, V., An Experimental Study on Validation Problems with Existing HTML Webpages, Proceedings of the 2005 International Conference on Internet Computing, June 27-30, Las Vegas, USA

[WDG] WDG HTML Validator, http://htmlhelp.com/tools/validator/

[CSE] CSE HTML Validator, http://www.htmlvalidator.com/

[Validome] Validome Validation Service, http://www.validome.org/

Acknowledgments

This paper is partially supported by Plovdiv University NPD grant NI2011-FMI-004.

Authors' Information

***Radoslav Radev** – Plovdiv University, Faculty of Mathematics and Informatics, 24, Tsar Asen Str., 4000 Plovdiv, Bulgaria; e-mail: radoslav_radev@gbg.bg*

Major Fields of Scientific Research: software engineering, object-relational mapping, aspect-oriented programming.