Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 11 (2013) 903 – 911

Procedia Technology

The 4th International Conference on Electrical Engineering and Informatics (ICEEI 2013)

A Metadata Approach for Building Web Application User Interface

Dimas Gilang Saputra, Fazat Nur Azizah*

School of Electrical Engineering and Informatics – Institut Teknologi Bandung, Jln. Ganesha no. 10, Bandung 40132, Indonesia

Abstract

In building a web application, a special attention is given on how the user interface is created. Although it seems that there are millions of ways in building web application user interfaces, experienced web application developers are usually familiar with particular "patterns". Nevertheless, user interfaces are often still developed from scratch, even if they are built based on existing applications. This makes web application user interface development a repetitive job. Furthermore, changes on the user interfaces often require a lot of work. This paper explores metadata as an approach to store the elements of user interfaces, so that the elements can be managed dynamically without having to go through the codes. We provide a preliminary metadata model of user interface elements of web applications in an entity-relationship model. We develop our model based on the nature of tree structure of the HTML documents. The metadata model is thus stored the elements of an HTML document as well as how these elements are related to each other in a typical hierarchical (parent-child) form of relationship. The metadata model will be a part of a metadata-based web application user interface generator.

Selection and peer-review under responsibility of the Faculty of Information Science & Technology, Universiti Kebangsaan Malaysia.

Keywords: CSS; generator; HTML; metadata; tree structure; user interface; web application

1. Introduction

The number of Internet users is growing rapidly in the recent years. In 2005, there are only about 360 millions of Internet users worldwide. By 2012, the number has reached 2.4 billion [1]. According to a 2011 survey by Cisco,

* Corresponding author. Tel.: +62-22-2508135; fax: +62-22-2500940.
*E-mail address:*13509038@std.stei.itb.ac.id

Selection and peer-review under responsibility of the Faculty of Information Science & Technology, Universiti Kebangsaan Malaysia.
doi:10.1016/j.protcy.2013.12.274

Internet has now become an important part of human life [2]. It is now an essential tool for communication, socialization, learning, finding information, shopping, and many more. One of the most prominent applications running on top of the Internet is the web application. Every day, millions of people access websites using various kinds of web browsers running on various kinds of hardware technology, including personal computers, mobile devices, even some home appliances, such as televisions and refrigerators. Some top websites include: Google (www.google.com), Facebook (www.facebook.com), and YouTube (www.youtube.com) [3].

In building a web application, a special attention is given on how the user interface is created. The user interface defines the interaction between people and the web application. It plays so important of a role that it may become the decisive point to define whether or not people revisit a website. With millions of web applications active on the Internet these days, people are getting accustomed to certain ways when interacting with the web applications. Although it seems that there are millions of ways in building the user interfaces, there are particular "patterns" that are followed by the applications, such as in how to log in/out, to insert new or to update data, and to show a table of data. Although experienced web application developers are usually familiar with these kinds of patterns, user interfaces are often still developed from scratch, even if they are built based on existing applications. This makes web application user interface development a repetitive job. Furthermore, when changes are required, modifications must be carried out on the codes of the application. Even a slight change in the user interface may introduce more time and money to be invested.

Nowadays, Internet users become more and more active and involved in the web applications. They wish not only to be able to enter and display their data, but also to be able to change the look and feel of the applications according to their personal preferences. This requires a special technique of programming the user interfaces which provides the opportunity for the user to do some customization.

Various programming techniques and tools are developed in order to overcome the problems. Template is a common technique that is used to provide some basic formatting on the user interfaces (usually based on HTML and CSS). The flexibility to do the formatting is usually limited and is still heavily decided by the developers. Application frameworks help developers to create web applications in shorter amount of time. While it may reduce the time to build an application, when changes are required on the application, the codes are still needed to be modified or new codes may have to be developed.

This paper explores the possibility to approach the aforementioned problems using metadata of the web application user interface. In this work, metadata is used to store the elements of web application user interface. The elements will then used to "generate" the actual code for the user interface. Any modifications and additions to the user interface of web applications are managed in the metadata. It is expected that this approach will overcome the addressed problems: instead of repeatedly building the same kind of programs for different web application user interfaces, there will be only one program which will work on different sets of metadata. Therefore, each web application user interface will have its own metadata. When changes are required, modifications are required only to the metadata, not to the entire program. Since metadata is also data, it is easier to manage the elements of the user interface stored in the metadata in comparison to manage source codes of web application programs.

2. Foundations and Related Works

2.1. Metadata

In Greek, *meta* means "about". Literally, metadata is data about data. NISO defines metadata as a "structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource" [4]. Tannenbaum defines metadata as "detailed description of the instance data; the format and characteristics of populated instance data; instances and values dependent on the role of metadata recipient" with instance data is defined as the input for receiving tool, application, database, or simple processing engine [5].

Three main types of metadata are [4]:

1. Descriptive metadata: resources intended for the discovery and identification data.
2. Structural metadata: shows how some of the objects collected, for example: a collection of pages sorted by chapter.

3. Administrative metadata: provides information to help manage resources, such as when and how it is created, file type and other technical information, and who has access to this information.

Metadata is used as a term for describing formal schemes of resource description by [4]:

- Providing resources retrieval by entering the keyword criteria (searching).
- Providing identification of the resources.
- Grouping the resources according to some characteristics.
- Distinguishing different resources.
- Providing information on the location of the resources.

Hyper Text Markup Language (HTML) format is often used to define a large variety of metadata, from basic data, such as texts and dates, to advanced metadata schemes, such as Dublin Core (www.dublincore.org) and e-GMS (www.esd.org.uk/standards/egms/). In the Structured Query Language (SQL), ANSI defines a standard metadata called the information schema in which the names, size, and other properties of databases, tables, columns, foreign key references, data types, etc. are stored. The standard is implemented in major database management systems, such as MSSQL, Oracle, MySQL, and PostgreSQL. An important use of metadata is in application generation area. In application generation approach, applications are not programmed from scratch, but are "generated" usually based on a particular metadata stored in a database.

2.2. Web Application

A web application is accessed by clients through the Internet or intranet [6]. A client requests a web application to a server and then the server replies by sending the web documents to the web browser in the client side. The communication between a web browser in the client side and a web server uses standardized protocol, usually the Hyper Text Transfer Protocol (HTTP). The architecture of web application can be observed in Fig 1. The web server, application server, file system, data, and external systems are located in server side machine(s). Web browser, on the other hand, is located in the client side machine(s).

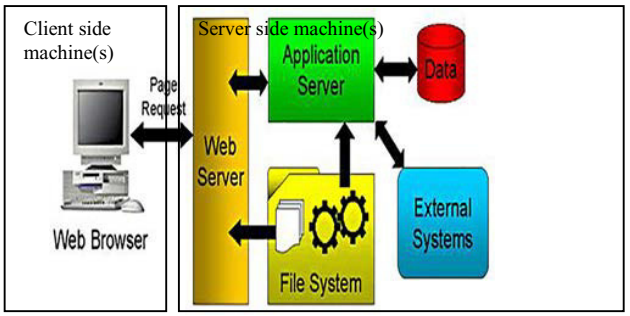

Fig 1. Web application architecture (from [7] with some adaptations)

The web server enables HTTP access to a website (the location in which web applications are resided) [8]. It processes the HTTP request and in return, provides the correct HTTP response. This is basically what happens when a web page is requested. Two types of web pages are defined: the static page and the dynamic one. When a static page is accessed, the web server searches through the file system to find the appropriate file as stated in the Uniform Resource Locator (URL) sent with the HTTP request, and then it sends back the appropriate HTTP response. When a dynamic page is requested, the web server must take an explicit programmatic action to generate the HTTP response, such as the execution of an application program, the inclusion of information from a secondary file, or the interpretation of a template. This requires the role of an application server.

The application server interacts with the web server in receiving requests, processing, and then giving back the responses. Some example of application servers are PHP, Java application server, and Microsoft .NET framework. The application server interacts with the file system in order to retrieve and modify the required files. It also

retrieves and modifies data in the databases. When necessary, it may also need to interact with other external systems, using technology such as web service.

2.3. HTML and CSS

A web application user interface is built from web pages in the form of HTML (Hyper Text Markup Language) format. An HTML document contains texts of a collection of tags. These tags define the set of elements of the web page. Web browsers are responsible to process the HTML documents, transforming the tags into the nice user interface that we see on the devices. HTML standards are set by the World Wide Web Consortium (W3C) standards with the latest one being the HTML5 [9].

The elements of web application user interface are indicated by HTML tags. These tags are written in angled brackets, for example: `<html>`. The element type's label and other attributes are defined inside the brackets. The element type label must be placed immediately after the opening angled bracket. In `<html>` tag, the element type label is `html`. Other attributes are optional and can be written in a sequence and in no particular ordering. A defined attribute must have its value and is written as `attribute="value"`. For example: `<html id="abc" class="xyz">`. Some tags have pairs with the closing pair is started with the `</` bracket. For example: `</html>` is the closing tag for `<html>`.

Tags can be place next to each other in a neighborhood form or within a hierarchy in which one element is placed underneath (as a child of) another element. For example: the tags `<h1>` and `<p>` in: `<h1>HeaderText</h1><p>ParagraphText</p>` are in neighborhood form, whereas the tags `<form>` and `<textarea>` in: `<form><textarea>Textarea-Text</textarea></form>` are in a hierarchical form. In the latter example, the `<form>` element is the parent of the `<textarea>` element or, in other words, `<textarea>` element is the child of the `<form>` element. The basic form of an HTML document is a hierarchical form with an `<html>` tag as the root tag.

Cascading Style Sheets (CSS) is a language that can be used to set the styles (colour, size, and so forth) of structured documents, such as HTML or XML (eXtensible Markup Language). Just as HTML, CSS also has been standardized by W3C [10]. Using the CSS, it is possible to separate the aspect of element styles and the content to be displayed. In a HTML document, CSS can be written as a `<style>` element within the `<head>` element. It can also be written in a different file called in the HTML document.

2.4. Related Works

This is not the first work that uses metadata to store elements of user interface. Microsoft .NET framework provides Windows Forms from metadata stored in an XML format [11]. The metadata is not only used to store user interface elements, but also client-side logic. In the .NET framework, some static elements of the user interface are required to be hard coded, while the metadata keep the customizable elements, such as screen layout, controls placed on the screen, their properties, and the position.

Metadata is also used in reporting engine, such as JasperReports (www.jaspersoft.com) and Pentaho Reporting (reporting.pentaho.com) to generate reports dynamically. In JasperReports, metadata is used in the form of a template in JRXML format to store the layout of a report. JRXML format is a special XML based format used only by JasperReports. In the metadata, elements of reports such as header, footer, and table, are recorded. Based on the metadata, reports can be generated in various formats, including HTML, Microsoft Excel, and PDF.

Our work focuses on the web application user interface and we strictly make use of the nature of the HTML and CSS formats. Furthermore, we use relational database to store the metadata of the user interface. Thus, we can create a metadata management application in the same way we create other typical three-tier web application.

As in other works, we separate the concern of user interface with the storage of data. We recognized that there are two types of data displayed in the user interface elements: the static and dynamic data. We take both into consideration in our metadata model.

3. The Tree Nature of an HTML Document

The elements (tags) of an HTML documents form a tree structure. Each of the elements can be seen as a node of the tree, forming a hierarchical structure. Among the elements with the same parent, neighborhood relationships are formed. This nature of HTML structure is exploited in this research as the basic approach in modeling the metadata of web application user interface.

An example of a simple web page can be observed in Fig 2 and the HTML code containing the elements that form the web page are shown in Fig 3. The `<html>` element is usually the root element with all other elements as its descendants, notably the `<head>` and `<body>` elements as its direct children. The `<head>` element contains the collection of metadata information of the HTML document, such as the title and the document character set. The `<body>` element contains the actual content of the document in which the eye-visible elements (the user interface elements) are placed. In a standard HTML document, only one `<html>` tag, one `<head>` tag, and one `<body>` tag are usually found. To help the browsers render the document correctly, a `DOCTYPE` element is sometimes placed on top of all other elements, including the `html` element.

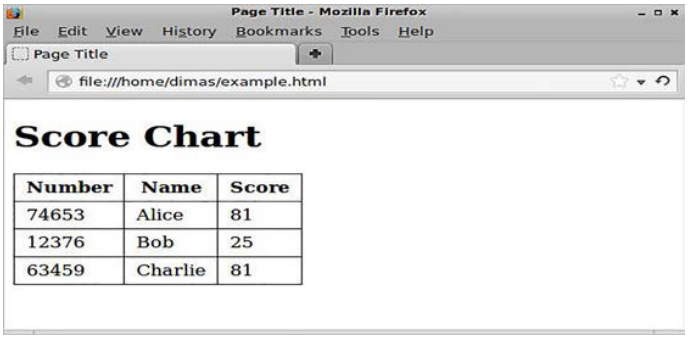

Fig 2. HTML page example

As shown in Fig 2 and Fig 3, the HTML document consists of one `<html>` tag with the `<head>` and `<body>` tags as its direct descendants. The `<title>` tag can be found inside the `<head>` tag in which the document title (shown in the browser window's title) is resided. The `<body>` tag hosts the contents shown in the web page, including an `<h1>` element and a `<table>` element. The `<h1>` tag contains a text meant to be caption of the succeeding table. Under the `<table>` tag, `<tr>` elements are defined as the table rows and within each `<tr>`, detailed cells are defined inside the `<th>` and `<td>` elements. The `<th>` elements define the columns for the header of the table, while the columns that hold the content of the table are defined by the `<td>` elements.

A visualization of the corresponding tree structure representation of the HTML elements of the document is depicted in Fig 4. Note that not every aspect of the HTML elements is shown in the visualization for the sake of simplicity, including the values held by each of the elements. This shows that the same structure can hold different values. Therefore, we can separate the values (aka. data) from the structure that holds them. As shown in Fig 2, the structure should hold the values of a score chart of 3 persons which most likely are retrieved from a certain database. Using other databases, the structure can hold other values, such as the list of patients or employees of a company.

4. The Preliminary Metadata Model of A Web Application User Interface Elements

The elements of an HTML document form a tree structure. This structure shapes the basis of our metadata model of web application user interface elements. In this research, the relational database is used to store the elements of the user interface. Therefore, for the conceptual model of the metadata, we use entity-relational model [12].

Fig 5 shows a simplified version of the entity-relational diagram depicting the essential parts of the metadata model. The model is still in the preliminary stage of development, showing only how the user interface elements of a web application are modeled. The model has not yet incorporated the storage of more than one web page in one

web application or more than one web application stored in the same server. It has neither dealt with the users who are allowed to create/make changes on the user interface elements. Nevertheless, it still provides the most essential part of the modeling on which other issues can be taken care of in the future.

Fig 3. An HTML code example

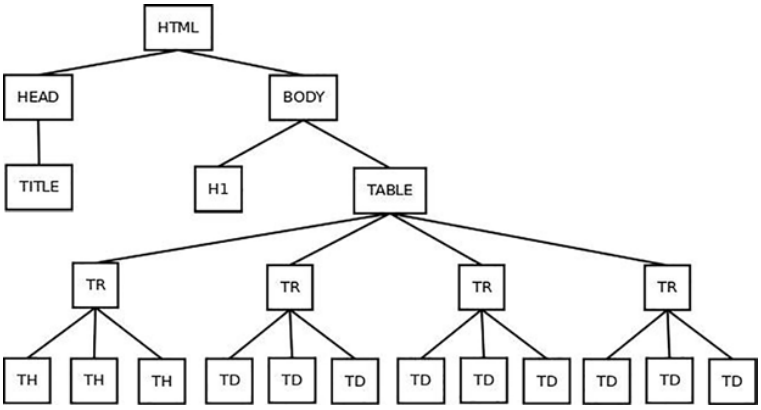

Fig 4. The corresponding HTML tree structure for the HTML code in Fig 3

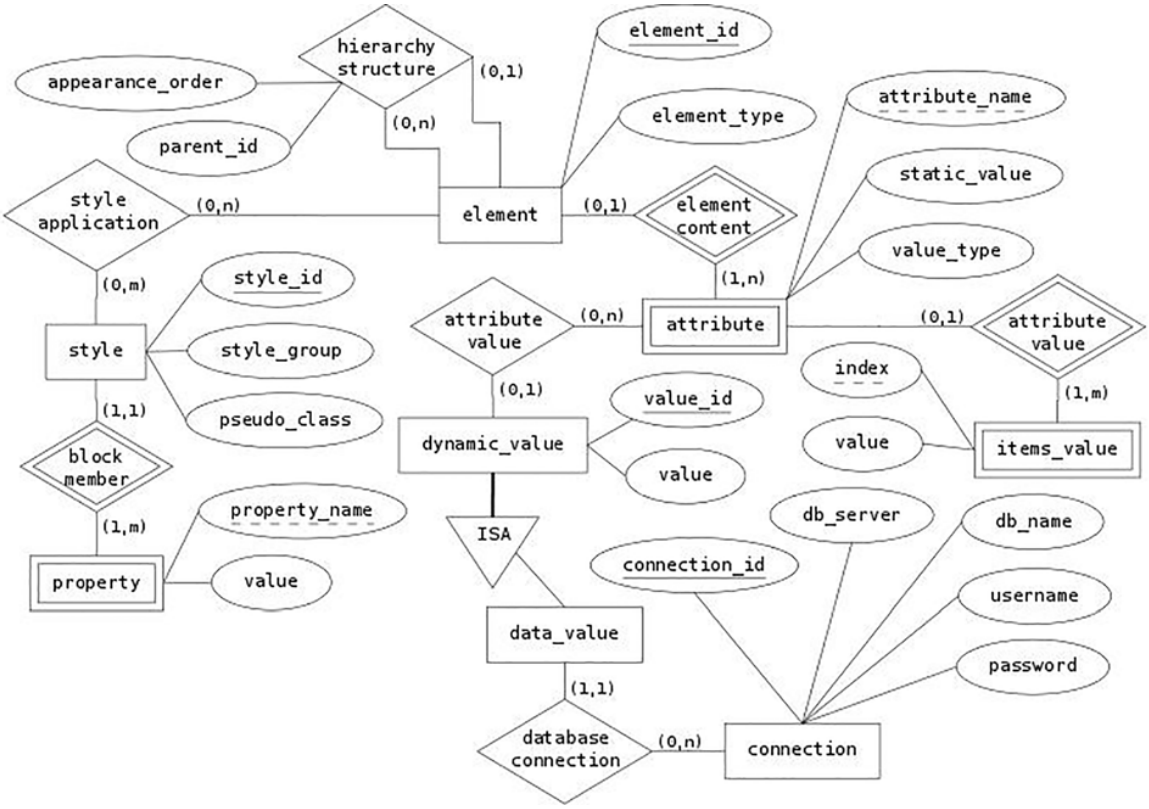

Fig 5. Entity-relationship diagram of the metadata model

The `element` entity type represents the elements of the user interface, including the attributes, such as ID (`element_id`) and the type of the element (`element_type`, representing the tags such as paragraph `<p>`, heading `<h1>`, image `<image>`, etc.). `Element` has a relationship with itself: `hierarchy_structure`, representing a hierarchical structure between one and another. The attribute `appearance_order` provides the order in which elements are arranged underneath the same parent element.

The elements make use of CSS to control how they are displayed. A CSS contains one or more *style blocks*. Each style block contains a *selector* and one or more *properties*. One style block can be used by one or more elements. Elements use selector to refer to a style. There are several *types of selector*, such as ID, class, attribute, child, descendant, type, or universal selector [13]. The trickiest part of the modeling of CSS comes in the modeling of the selector because of its relatively free-form nature. In our model, we group each type of selectors into *style groups*. The style group name is used as a class name when an element needs to use the respective a style block. One style block can have different style when an element is in different states. The states in which an element can be in are called *pseudo-class* [13]. Examples of the states are: when an element is being hovered (`:hover`), when the link of the element is visited (`:visited`), and when an element is the first child of its parent (`:first-child`).

In our model, the `style` entity type has style group name (`style_group`), property (`property_name`), and the element state in which it can be applied to (`pseudo_class`) as its attributes. The `element` entity and `style` entity share many-to-many relationship to show that one style block can be used in more than one element and vice versa. Each block style can have one or more properties. In our model, style `property` is modeled as a weak entity related to `style`.

As stated earlier, we separate the structure of a user interface with the data that must be displayed. The HTML elements and the CSS are part of the structure of the user interface. The data, on the other hand, are held by the

attributes of an element. An element can have one or more attributes. In our model, `attribute` is a weak entity related to the `element` entity type with its name (`attribute_name`), static value (`static_value`), and `value_type` as its attributes. Static value (`static_value`) contains the value of an attribute that will never be changed. Value type (`value_type`) provides other types of value that can be placed in the attribute. There are three types of such value: the repetitive items, the text value, and data fetched from database. The repetitive items are static repetitive values stored in the `items_value` entity type. The text value and the data from the database are together considered as dynamic data because they change according to particular parameter provided by the program. Thus, they are modeled into a generic entity called `dynamic_value`. A specialized version of the `dynamic_value` is the `data_value` entity type which models the data fetched from the databases. It has a special relationship to represent the connections to the databases. When a `dynamic_value` is a text value, the `value` attribute of the entity type contains a text as the value. When it is data from database, the `value` attribute contains an SQL statement.

In building a web application user interface, it is important for the developers not to reinvent the wheel. We use *templates* as a way to provide developers with ready-to-customize skeletons of application. We provide structures commonly found in a lot of web applications, such as the structures of Create-Read-Update-Delete (CRUD) forms/reports and let the developers customize some parts of the structures. Thus, the developers do not have to create everything from scratch. We argue that most web applications still require those common forms/reports/pages and they have general structures that web application users are familiar with.

Essentially, there is no difference between the structure of a template and the structure of a web application. We model the template just as an ordinary web application, only without values/data attached to the structure. Therefore, the separation between the structure and data takes an important role here.

Fig 6 shows how we model the template. `template` entity type represents a template which attaches to a particular `element` as the root element of the template. The rest of the structure of the template is handled by the hierarchical structure of `element` entity type. Template has an ID (`template_id`) as its attribute.

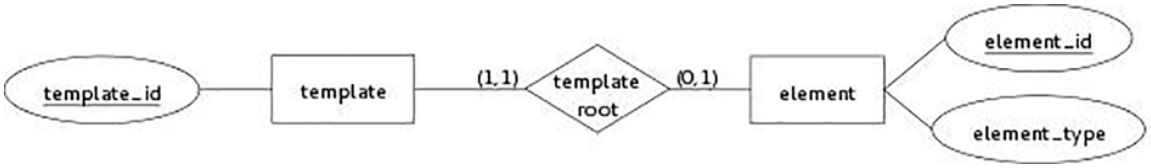

Fig 6. Entity-relationship diagram for template

4. The Metadata-Based Web Application User Interface Generator

Fig 7 shows the architecture of the future metadata-based web application user interface generator in which our metadata model will be applied. The *UI metadata editor* component works as the tool for users to manipulate the user interface metadata (*UI metadata*). This allows people to edit the user interface elements, without having to deal directly with the codes of the program. The *UI generator* component combines the user interface metadata and the data (stored in separate databases) to generate the pages of web application.

5. Conclusions and Future Works

In this paper, we present a preliminary metadata model of user interface elements of a web application. We develop our model based on the nature of tree structure of the HTML documents. The metadata model is thus stored the elements of an HTML document as well as how these elements are related to each other in a typical hierarchical (parent-child) form of relationship. The model has also covered the modeling of the attributes of the elements, as well as dealing with repeating items of a particular element. Styles can be applied on user interface element and therefore, the metadata model is also equipped with the possibility to include styles in the modeling. As a preliminary model, the metadata model has not yet covered several important issues, such as the handling of more

than one web applications in a single model as well as management of the users. It is planned that in the near future, the model will be improved with more features.

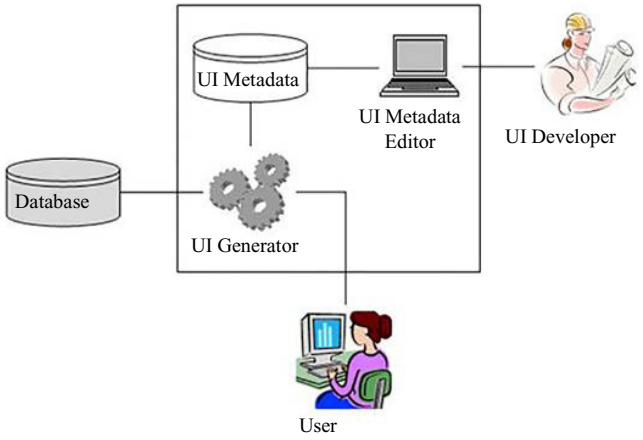

Fig 7. The architecture of metadata-based web application user interface generator

The metadata model is expected to be the foundation of a metadata-based web application user interface generator. The architecture of such generator has been described and using the improved metadata model, a prototype version of the generator will soon be developed. The relational database will still be the basic technology used to implement both the user interface metadata and the application data. This prototype are planned to be tested to build several typical web applications. The performance of the applications has not yet become our highest priority in this research. This research is aimed at proving whether our concept works or not. Other considerations will be the priority on our next researches.

References

[1] World Internet Users Statistics Usage and World Population Stats. http://www.internetworldstats.com/stats.htm. Retrieved 9 December 2012.
[2] 2011 Cisco Connected World Technology Report, Cisco, 2011.
[3] Alexa Top 500 Global Sites. http://www.alexa.com/topsites. Retrieved 10 December 2012.
[4] NISO. Understanding Metadata. NISO Press, Bethesda, MD. 2005.
[5] Tannenbaum, A. Metadata Solutions. Addison-Wesley, Upper Saddle River, NJ. 2001.
[6] Tanenbaum, A. S., & Steen, M. v. Distributed Systems: Principles and Paradigms, 2nd edition. Prentince Hall, Upper Saddle River, NJ. 2006.
[7] The Architecture of Web Application. http://www.ibm.com/developerworks/ibm/library/it-booch_web/. Retrieved 28 February 2013.
[8] Shklar, L. & Rosen, R. Web Application Architecture. John Wiley & Sons, West Sussex, England. 2003.
[9] HTML5. http://dev.w3.org/html5/spec. Retrieved 30 November 2012.
[10] Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. http://www.w3.org/TR/2011/REC-CSS2-20110607/. Retrieved 25 October 2012.
[11] Metadata-Driven User Interfaces. http://msdn.microsoft.com/en-us/library/ms954610.aspx. Retrieved 13 May 2013.
[12] Silberschatz, A., Korth, H. F., & Sudarshan, S. Database System Concepts. Fourth Edition. McGraw Hill. 2002.
[13] CSS Selectors. http://www.w3.org/TR/CSS2/selector.html. Retrieved 28 May 2013.