www.calsec.or.kr
http://dx.doi.org/10.7838/jsebs.2013.18.3.143

HTML5 문서로부터 OWL 온톨로지 구축 기법

A Conversion from HTML5 to OWL Ontology

손태모(Taimao Sun)*, 윤이연(Yiyeon Yoon)**, 김우주(Wooju Kim)***

초 록

웹의 발전으로 말미암아 새롭게 부상하고 있는 차세대 웹 언어인 HTML5의 웹 기술 언어 표준화가 현재 진행 중이다. 웹의 관점에서 볼 때 HTML5에 새롭게 추가된 구조적 시맨틱 요소들로 인하여 예전보다 웹의 기술에서 웹에서의 데이터들을 더욱 의미적으로 표현할 수 있게 해주는 것이 가능하게 되었다. 이는 또한 새로운 HTML5로 기술된 웹 문서에서 유용한 정보를 쉽게 추출할 수 있음을 나타내기도 한다. 따라서 웹 문서에 새롭게 추가된 시맨틱 요소들에 대한 온톨로지화를 통하여 웹 문서 정보에 대한 직관적이고 구조적인 접근방식이 필요하다. 본 연구에서는 HTML5에 새로 추가된 시맨틱 요소들에 입각하여 웹 문서 요소들의 온톨로지 모델과 인스턴스들의 매핑 방법론을 통하여 효과적인 HTML5 문서의 온톨로지화 방법론을 제안하고자 한다.

ABSTRACT

HTML5, new standard for web language, is being standardized corresponding to the development of web. Since several new semantic elements have been added into HTML5 standard, current Web Environment is becoming more and more semantic. In order to provide better user experience by using information extraction from HTML5 page, new HTML5 Elements should be mapped to a corresponding Ontology. In this research, we will focus on new semantic elements to build Ontology from HTML5 document. For this purpose, we will propose a methodology of Schema level mapping rule and instance mapping rule.

키워드 : HTML5, OWL, 온톨로지, 온톨로지 매핑, 시맨틱 웹
HTML5, OWL, Ontology, Ontology mapping, Semantic Web

본 논문은 중소기업청에서 지원하는 2012년도 산학연협력 기업부설연구소 지원사업(No. 00045640)의 연구수행으로 인한 결과물임을 밝힙니다.

* Department of Information and Industrial Engineering, Yonsei University
** Department of Information and Industrial Engineering, Yonsei University
*** Corresponding Author, Department of Information and Industrial Engineering, Yonsei University (E-mail : wkim@yonsei.ac.kr)

2013년 05월 31일 접수, 2013년 07월 17일 심사완료 후 2013년 08월 06일 게재확정.

1. 서 론

웹의 발전으로 인해 전 세계적으로 많은 사람들이 정보수집과 공유의 도구로 웹을 사용함에 따라 웹에는 끊임없이 데이터가 생성되고 있다. 하지만 이렇게 생성된 데이터들은 지금의 웹 표준 언어인 HTML로 생성된 경우가 대부분이기 때문에, 기계가 인식 가능하게 하기보다는 사용자 용이성에 맞게 제작되어 있음으로 시맨틱 정보를 표현하기에는 많은 한계가 있었다. 데이터의 분산은 물론이고 문서 표현 자체도 비구조적인 형태를 가지고 있음으로 엄청난 양의 HTML 데이터를 활용하는데 큰 어려움이 있다. 이러한 빅 데이터의 시대에 웹에 분산되어 있는 데이터들을 활용하기 위해서는 데이터가 기계가 읽을 수 있는 언어로 표현되어야 하며, 따라서 이를 위해서는 시맨틱 요소가 더 반영되어야 한다. 팀 버너스 리가 제안한 미래의 웹 즉 시맨틱 웹은 바로 웹의 데이터들이 기계가 읽을 수 있는 언어로 표현된 웹이라 할 수 있다. 단순히 링크드 데이터의 개념 아니라, 기계가 데이터의 의미(semantics)를 이해함으로써 웹의 사용자들에게 지금보다 훨씬 더 향상된 웹 서비스를 제공하고자 하는 것이 시맨틱 웹이며 시맨틱 웹의 핵심기술로서 온톨로지 기술을 꼽을 수 있다. 온톨로지는 하나의 특정도메인에 대한 지식표현의 방식으로서 특정 도메인의 이해와 사용에 도움을 주는 개념화 명세서로 정의되고 있다. 온톨로지는 지식의 공유, 논리추론의 가능성, 지식 및 정보의 재사용성 등 다양한 이점을 가지고 있기에 대량의 웹 문서들을 온톨로지화 함으로 인하여 지능화된 시맨틱 웹에 더 가까워질 수 있다.

최근에 웹 언어가 HTML5로 진화하면서 가장 주목할 바는 바로 HTML5에 시맨틱 웹 개념의 구조적 요소들이 추가되었고 또한 텍스트 본문에 대해서도 시맨틱 Annotation을 활용하여 태깅 함으로써 텍스트 본문에 대하여 좀 더 쉽게 온톨로지화 할 수 있는 기법이 포함되어 웹 소스코드들이 이전보다 더 의미를 가지게 되었다는 점이다. 기존의 HTML 4.01 환경에서의 HTML 문서에 대한 온톨로지화는 테이블이나 INPUT FORM 요소들만을 대상으로 시도하였거나, 혹은 HTML 4.01자체의 문서 구조를 나타낼 수 있는 데 선천적인 결함(예 : div요소의 남발)이 있기 때문에 HTML5 문서를 온톨로지화 하기에는 부족한 점이 많다. 아울러 HTML5 문서 온톨로지화에 대한 연구도 아직 많이 미흡한 실정이다.

따라서 본 연구에서는 HTML5에 새롭게 추가된 시맨틱 요소들에 입각하여 HTML5를 온톨로지화 하는 방법론을 제안하고자 한다.

본 연구의 구성은 다음과 같다. 먼저, 제 2장에서는 시맨틱 웹과 그 기술, 온톨로지 및 HTML 문서의 온톨로지화에 대한 관련 연구 동향을 살펴보고, 제 3장에서는 온톨로지화의 방법론에 대하여 설명한다. 여기서는 실제 HTML5 문서 구조를 통해 OWL 모델을 생성하는 방법과, 생성된 OWL 모델에 인스턴스를 추가하는 내용으로 구성되어있다. 제 4장에서는 결론 및 향후 연구 방향에 대한 논의로 이루어진다.

2. 관련 연구

2.1 시맨틱 웹(Semantic Web)

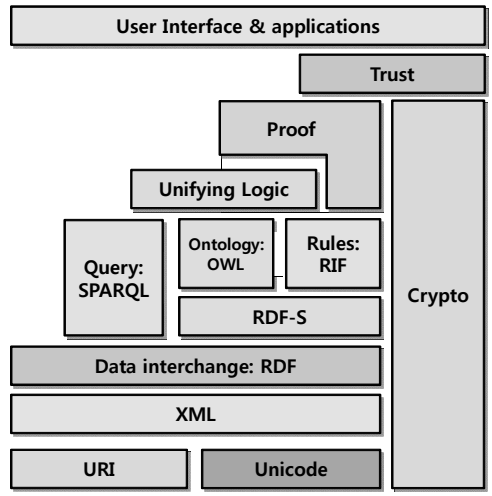
시맨틱 웹은 웹의 창시자인 팀 버너스리에

의하여 1998년에 제안된 차세대 웹 환경 기술이다. 시맨틱 웹은 웹에 분산된 여러 가지 리소스들 하나하나를 자원으로 보고 그 자원들 사이의 관계를 정의함으로써 컴퓨터가 분석하고 처리할 수 있는 형태로 표현하여 컴퓨터의 자동처리가 가능하게 하는 것을 목적으로 하여 탄생 되었다. 따라서 시맨틱 웹은 웹에 존재하는 자원들(이미지, 텍스트 등등)에 관계를 통한 의미를 부여함으로써 컴퓨터가 해당 자원의 의미 해석이 가능하며, 나아가 컴퓨터의 자동 커뮤니케이션이 가능하도록 하여 기계가 이해한 정보를 사용자들에게 서비스를 제공하는 것이 가능한 차세대 웹 환경이라고 볼 수 있다. 이는 현재 웹을 구성하는 HTML(Hyper Text Markup Language)로 기술된 웹 페이지들이 단순히 사람에게 보여주기 용이한(Human Readable) 목적으로 구성되어 있는 것과 비교할 때에 팀 버너스 리가 웹 개념을 제안할 때의 청사진에 더 접근한 모습이라고 볼 수 있다.

2.1.1 시맨틱 웹 기술 요소

현재 여러 국제기구 및 조직들에서 시맨틱 웹에 관련한 연구가 진행 중인데 현재 W3C의 주도 하에 표준화 사업이 진행 중이다. <Figure 1>은 W3C에서 제시하는 시맨틱 웹의 전체 프레임워크를 나타낸다[10].

웹에서 특정 자원들에 의미를 부여하기 위해서는 해당 자원을 식별하기 위한 방법론으로 URI이 제안 및 사용되고 있다. URI는 Uniform Resource Identifier)의 약자로 인터넷에 있는 자원을 나타내는 유일한 주소로 사용되며 URL(Uniform Resource Locator)과 URN(Uniform Resource Name)을 포함한다. 시맨틱 웹은 RDF(Resource Description Framework)에 기반하며 나타내고자



〈Figure 1〉 Semantic Web Structure

하는 컨셉을 주어(Subject), 술어(Predicate), 목적어(Object)의 트리플(Triple)로 표현함으로써 Conceptualization의 가장 작은 단위로 볼 수 있다. 즉 트리플로 최소한의 의미를 나타낼 수 있다. 여기서 주어, 술어, 목적어는 각각 URI로 표현되며 인터넷 환경에서의 RDF 언어를 표현하기 위하여 XML 마크업 언어를 사용한다. XML은 인터넷 환경에서 구조적 문서의 전송을 위하여 제안된 마크업 표준 언어로서 RDF의 문서표현 방식으로 사용되고 있다. RDF-Schema는 동일한 특징(Property로 표현됨)을 가지고 있는 여러 요소들의 집합과 그 집합들 사이의 관계를 표현하는 의미론적 어휘로서 데이터 모델의 제한되고 간단한 의미론만 제공한다. RDF-Schema의 이러한 표현의 단순함을 보완하여 더욱 풍부한 의미론을 제공하기 위한 추가적인 대안으로 OWL (Ontology Web Language)가 제안되었다. OWL은 RDF-Schema의 기초 위에 더 풍부한 의미표현 프로퍼티와 클래스들, Disjointness와 같은

클래스들 사이의 관계, Cardinality, Equality 등을 제공한다.

2.1.2 온톨로지

온톨로지는 시맨틱 웹을 구현할 수 있는 도구로서 지식 개념을 의미적으로 연결할 수 있는 도구이다[3]. 가장 많이 인용되는 온톨로지에 대한 개념적인 접근으로는 Tom Gruber가 1993년에 정의한 것이다. 이는, 리소스들의 공통점을 추출하고 개념화 하여 하나의 집합으로 정의하기 위한 표현(Description) 혹은 명시(Specification)이라고 볼 수 있다. 온톨로지를 구축함으로써 명시된 도메인에서의 지식 표현이 가능하게 된다. 대표적인 온톨로지 언어로는 RDF, OWL, SWRL 등이 있다.

2.2 HTML5

HTML5은 현재 사용하고 있는 W3C HTML 표준인 (X)HTML 4.01의 후속작으로 웹 환경에서의 차세대 마크업 언어이다.

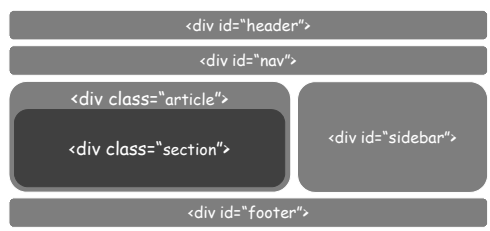
2.2.1 HTML5 개요

웹이 발전함에 따라 웹 브라우저 전쟁에서 비표준 요소들이 남발하게 되었고 또한 2000년대 중반에 웹 2.0 시대에 들어서면서 구조, 표현 및 데이터를 분리한 웹 표준 기법과 플랫폼으로서의 웹을 가능하게 한 Ajax 등 기술의 출현으로 말미암아 웹은 단순히 텍스트의 컨테이너로서 표현의 역할만이 아니라 데스크 탑의 소프트웨어와 같은 사용자 경험을 제공할 수 있게 되었다. 하지만 W3C는 웹의 이런 변화를 수용할 준비를 하지 못함으로 인해 모질라, 애플, 오페라 등 웹 브라우저 기업들은 2006년 W3C 밖에서 새로운 하이퍼 텍스트 워킹 그룹(Web Hypertext Application Technology Working Group, WHATWG)를 결성하여 새로운 표준안을 만들어냈다. 그들의 핵심은 HTML이 가진 가치를 인정하고 웹 브라우저 간 명확하지 않은 처리 방식에 대하여 재정의하고 새로운 마크업과 API를 통하여 Front End 웹 개발자들이 콘텐츠 중심의 어플리케이션 개발을 용이하게 하려는 것이다[11].

HTML5에는 HTML 4.01에 비해서 많은 새로운 기능들이 추가 되었는데, 시맨틱 웹 관점에서 봤을 때 중요한 부분은 구조적 마크업 요소들, 및 progress, time 등 의미기반 요소들의 추가와 RDFa, Microdata로 대표하는 콘텐츠에 대한 시맨틱 표현의 추가이다.

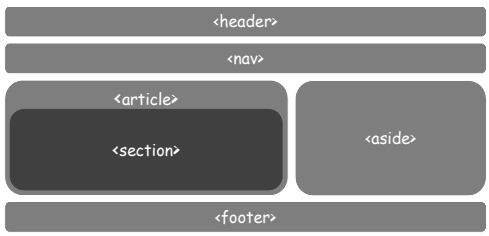
대부분의 HTML 4.01 웹 페이지들은 구조적 정의 마크업 요소의 결여로 header, footer column 등을 표현하기 위해 div 요소에 id나 class 속성을 이용하여 그 구조를 표현하는 것이 비공식적으로 일반화되었다. 비록 div 요소로 표현의 목적에 달성할 수 있지만 하나의 웹 페이지에 div 요소가 무분별하게 남발하는 단점이 있어 명확한 의미론적 구조를 나타내지 못하였다.

따라서 HTML5는 이와 같은 단점을 보완하여



〈Figure 2〉 HTML4 Structure Represention

여 개발되었고, 새로 추가된 요소들인 header, nav, article, section, aside, footer 등으로 더욱 의미론적인 sectioning 방식을 제공할 수 있다.



〈Figure 3〉 HTML5 Structure Representation

2.3 HTML의 온톨로지화 관련 기존 연구

web 1.0 web 2.0 시대를 거치면서 웹은 하나의 서비스 제공자가 제공하는 서비스를 수많은 사용자들이 사용하는 방식으로부터, 모든 사용자들이 정보의 사용자이면서 또 정보의 제공자가 되는 방식으로 발전하였다. 이러한 웹 서비스의 방식의 변화는 웹에서의 정보양의 대폭적인 증가를 초래하였다. 특히 SNS의 출현은 정보홍수의 시대에 들어서는 계기가 되었다. 하지만 이러한 방식으로 생성된 많은 양의 정보는 대부분이 비정형 데이터들로써 정보의 발견과 재사용이 어렵다는 단점이 있다. 따라서 이와 같은 비정형데이터들에 의미의 부여함으로써 정형화된 데이터로 변환하고, 이를 통해 비정형 데이터들로부터 정보를 추출하여 활용하기 위한 연구들이 진행되어 왔다. 이 중 웹 데이터의 온톨로지화는 이러한 비정형 데이터를 의미 있는 데이터로 변환하기 위한 방법으로 간주되어 그 동안 많은 연구가 진행되어 왔다. 따라서 본 연구에서는 HTML을 시맨틱 웹 환경에서도 사용 가능하게 하기 위한 HTML 온톨로지화 관련 기존 연구 분석을 위해, HTML 문서로부터 RDF 추출 연구와, HTML 문서를 통해 OWL로 변환하는 두 가지 방법으로 구분하였다. 이는 HTML 문서를 변환하는 언어에 따라 구분한 것으로, 다음 <Table 1>는 이와 같은 기준으로 HTML의 온톨로지화 관련 연구에 대한 내용을 정리한 것을 나타낸다.

〈Table 1〉 Previous Research

	Researcher	Research Contents
RDF	Kim et al. [6]	HTML Document Metadata Extraction and DC Metadata based RDF Metadata Document Generation
	Lee et al. [8]	RDF Ontologization of HTML Metadata
OWL	Hwangbo and Lee[2]	Ontologization based on HTML Element(s) Classification
	Kwak et al.[7]	OWL Syntax Definition based on HTML Elements Structural Semantics

각각의 논문을 살펴보면, 먼저 김기면 외[6]는 웹 페이지에서 더블린 코어의 엘리먼트에 해당되는 메타데이터를 자동 추출하여 RDF를 이용한 RDF 메타데이터 자동생성 편집기에 관한 연구를 진행했다. 이는 HTML 페이지의 head에 포함되는 메타데이터의 추출 및 온톨로지화에 초점을 맞추고 있다.

이미경 외[8]는 HTML로 기술된 웹 사이트 정보를 RDF 데이터 모델링 기법을 이용하여 RDF 메타데이터로 자동 생성하고 필요한 경우 추가 정보를 입력하여 메타데이터를 재 생성하는 새로운 시스템을 제안하였다.

Hwangbo and Lee[2]는 먼저 모든 HTML 요소들을 Title, Metadata, Link, Table, List, Paragraph, Text, Image, Form 등 9개 집합으로 분류하고, HTML 문서 내의 정보를 Text based tag와 Non text based tag로 분류하여 각각의 요소들의 온톨로지 매핑 방법론을 정의하였으며, 또한 텍스트 본문에 대해서도 Parts of Speech Tagging을 통한 주어-술어-목적어 트리플 추출을 시도하였다.

곽현수 외[7]는 HTML 웹 페이지 중 의미를 나타낼 수 있는 요소만을 분석하였다. 즉 Title, Table, input Form에서의 Select, Radio, Checkbox 등 6개의 요소들에 대한 정보들만 추출하여 온톨로지화 하였다.

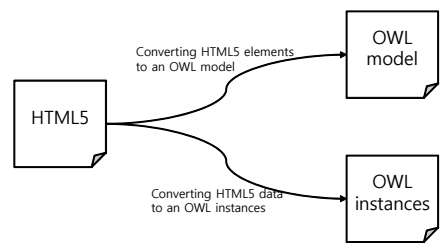
Brian et al.[1]은 HTML5/CSS3 웹 기술을 이용하여 의료 이미지에 Annotation을 하는 방법론을 제안하였다. HTML5에 새롭게 추가된 캔버스에 기초하여 이미지에서의 원이나 점을 선택하고 온톨로지에서 해당 선택 부분과 매핑되는 인스턴스로 Annotation해주는 방법론이다.

이상 관련연구를 분석한 결과, HTML을 온톨로지화 하는 연구에서는 각 변환 언어에 상관없이 트리플 형태로 표현하면서 온톨로지화 작업을 진행한 것을 볼 수 있다. 하지만 지금까지 진행된 연구들은 HTML에 있는 모든 요소를 온톨로지화 하기보다는 주로 head에 포함되는 메타데이터를 대상으로 온톨로지화 하는 수준의 연구가 진행되었으며, HTML을 온톨로지화 하더라도 HTML5에 새롭게 추가된 요소들에 대한 선행연구는 아직 미흡한 것으로 보인다.

또한 관련분야 선행연구를 살펴보면 온톨로지화 기법으로 RDF 언어를 사용한 연구와 OWL 언어를 사용한 연구로 나뉘고 있다. 우선 RDF 언어는 메타데이터를 정의하기 위한 개념적 구조를 제공하고 있지만 그 표현력에서 많은 제한이 있는 반면에, OWL 언어는 (비)동치성, 속성의 특성, 제약, 클래스의 공리, 조합 등 요소들이 강화되어 도메인의 지식의 공유와 재사용에서 RDF보다 풍부한 표현력을 제공한다. 따라서 본 연구에서는 온톨로지화 작업언어로 OWL 언어를 사용하여 HTML5에서 새롭게 추가된 요소에 대한 온톨로지화 작업과, head 뿐만 아니라 body요소 안에 있는 요소집합들에 대한 온톨로지화 방법론을 제안하고자 한다.

3. HTML5 2 OWL 방법론

본 연구에서는 HTML5 문서에 기반한 HTML5 2 OWL 방법론을 제안한다. <Figure 4>는 HTML5를 OWL 온톨로지로 변환하는 방법을 도식화한 것이다. 이 방법은 두 가지 단계로 구성되어 있으며, 첫 번째 단계는 HTML5 문서에 있는 엘리먼트를 통해 OWL 모델을 만들고, 그 다음으로는 HTML5 문서에 포함된 데이터를 통해 OWL 인스턴스를 채우는 방법으로 진행한다.

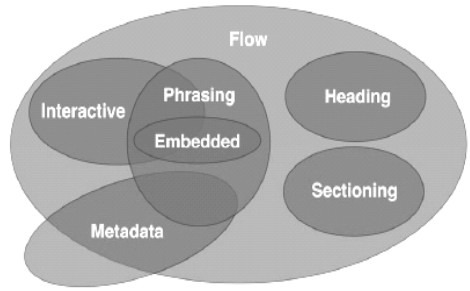


〈Figure 4〉 HTML to OWL Ontology Methodology

본 연구에서는 이와 같은 방법으로 HTML5 문서를 통해 OWL로 변환함으로써 HTML5을 온톨로지화 하고자 한다. 먼저 HTML5 문서로부터 OWL 모델을 생성하는 단계는 제 3.2절에서 기술하고, 이렇게 만들어진 OWL 모델에 인스턴스를 생성하는 단계는 제 3.3절에서 기술한다.

3.1 HTML Tag Set에 대한 정의

HTML5는 기존의 HTML 요소에서 시맨틱 요소를 포함한 새로운 요소들이 추가 되었다. 이에 따라 WHATWG에서는 <Figure 5>와 같이 HTML5의 각 요소 별로 새로운 특성에 따라 분류를 하였다. 그림에서 표현된 각각 요소들은 임의의 분류에 속하지 않을 수도 있고 또한 여러 분류에 동시에 종속될 수도 있다. 본 연구에서는 시맨틱 요소가 포함된 요소를 연구대상으로 함으로써, <Figure 5>의 Heading과 Sectioning 분류를 대상으로 HTML5을 OWL 온톨로지로 변환한다. 각각의 분류에 종속된 요소는 <Table 2>와 같다.



〈Figure 5〉 Contents Model

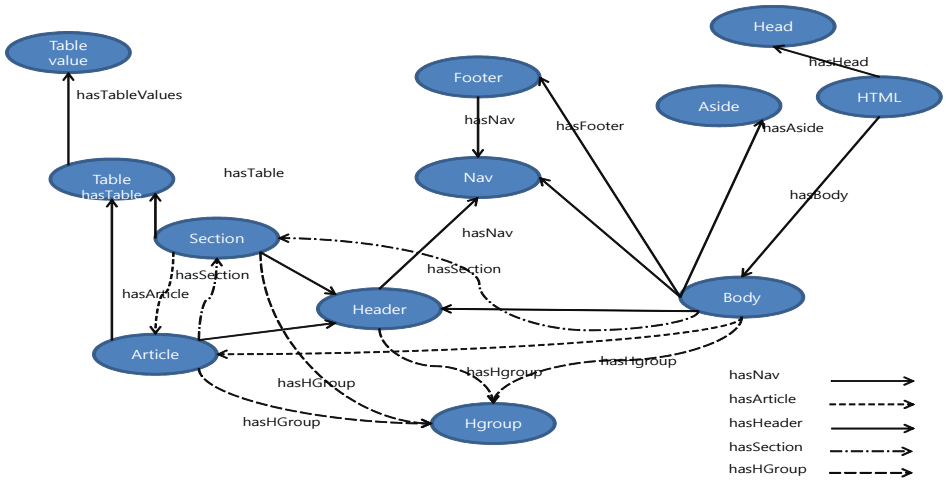
〈Table 2〉 HTML5 Semantic Elements

Class	Elements
Heading	article, aside, nav, section, header, footer
Sectioning	h1 ~ h6, hgroup

본 연구에서는 <Table 2>에 정의된 요소를 바탕으로 HTML5 to OWL 방법론을 제안한다.

3.2 HTML 스키마 정의 및 OWL 모델 생성

구조적인 HTML 요소를 통한 OWL 모델의



〈Figure 6〉 HTML5 Elements Structure

생성을 위해 해당 HTML 요소의 구조적 관계를 고려한 클래스 정의가 필요하다.

본 연구에서는 이들 요소에 대한 구조적 관계를 <Figure 6>과 같이 표현하였으며, 각 클래스에 대한 상세한 설명은 다음과 같다.

3.2.1 Header

header 요소는 header 클래스로 정의된다. 소개나 기능들의 묶음, 문서나 각 section, article 등의 header 부분에 해당되며 해당 부분들의 제목 혹은 간단한 소개를 포함할 수 있다.

h요소는 문서의 header line으로써, 제목의 타입을 정의하기 위해 header에 포함이 될 수 있으므로 header 클래스의 DatatypeProperty인 #hasHeaderType으로 정의하고, h요소가 보여주는 text도 DatatypeProperty인 #headerValue로 정의한다. 따라서 header 클래스는 <Figure 7>과 같이 정의된다.

```
<owl : Class rdf : ID="Header"/>
    <owl : DatatypeProperty rdf : ID="hasHeaderType">
        <rdf : type rdf : resource="http : //www.w3.org/2002/07/owl#
                FunctionalProperty"/>
        <rdfs : range rdf : resource="http : //www.w3.org/2001/XML
                Schema#int"/>
        <rdfs : domain rdf : resource="#Header"/>
    </owl : DatatypeProperty>
    <owl : DatatypeProperty rdf : ID="headerValue">
        <rdf : type rdf : resource="http : //www.w3.org/2002/07/owl#
            FunctionalProperty"/>
        <rdfs : domain rdf : resource="#Header"/>
        <rdfs : range rdf : resource="http : //www.w3.org/2001/XML
            Schema#string"/>
    </owl : DatatypeProperty>
```

〈Figure 7〉 Header Class

3.2.2 Nav

nav 요소는 글로벌 네비게이션을 위해서 구성된 섹션을 나타낸다. 즉 해당 웹 페이지의 주요 네비게이션 링크들을 묶어주는 역할을 한다. 한 페이지 내에서 nav 요소를 하나만 사용해야 하는 것은 아니나, 웹 페이지에서 구조적으로 볼 때에 nav는 body 요소 안에 종속됨으로 body 클래스가 nav 클래스를 가진다고 선언한다. 또한 원칙적으로 header 요소 안에 nav 요소를 삽입하여 사용할 수 있음으로 header 클래스도 nav 클래스를 가진다고 선언한다.

nav는 여러 웹 페이지들로 이동하는 하이퍼링크들의 List임으로 rdf : Bag 타입으로 정의한다. 따라서 nav클래스는 <Figure 8>과 같이 정의된다.

```
<owl : Class rdf : ID="Nav"/>
    <Body rdf : ID="bodyExample">
      <hasNav>
          <rdf : Bag rdf : ID="navigationExmaple">
            <rdf : li rdf : datatype="http : //www.w3.org/2001/XML
                Schema#string">About</rdf : li>
            <rdf : li rdf : datatype="http : //www.w3.org/2001/XML
                Schema#string">Contact</rdf : li>
            <rdf : li rdf : datatype="http : //www.w3.org/2001/XML
                Schema#string">Tutorial</rdf : li>
            <rdf : li rdf : datatype="http : //www.w3.org/2001/XML
                Schema#string">Home</rdf : li>
          <rdf : type rdf : resource="#Nav"/>
          </rdf : Bag>
      </hasNav>
    </Body>
```

〈Figure 8〉 Nav Class

3.2.3 Aside

aside는 본문과 직접적인 연관성이 적은 내용들을 표시하는 색션이다. 본문과 관련이 없는 사이트나 광고 등을 기술하는 데 사용되며 주로 사이드바로 활용된다.

aside 요소는 내용과 관련이 없는 내용을 적거나 다른 사이트로의 링크 혹은 광고 등 목적으로 사용할 수 있다. 따라서 단순히 텍스트를 기술할 경우의 DatatypeProperty로 #hasValue

를 가지며 또한 다른 사이트로의 링크를 리스트로 가질 수 있기 때문에 또한 rdf : Bag 타입으로 선언한다. aside는 body 요소 안에 기술됨으로 body 클래스가 aside 클래스를 #hasAside 한다고 ObjectProperty를 정의한다. 그러므로 <Figure 9>와 같이 aside클래스를 정의한다.

```
<owl : Class rdf : ID="Body"/>
<owl : Class rdf : ID="Aside"/>
<owl : ObjectProperty rdf : ID="hasAside">
    <rdfs : range rdf : resource="#Aside"/>
    <rdfs : domain rdf : resource="#Body"/>
</owl : ObjectProperty>
<owl : DatatypeProperty rdf : ID="hasValue">
    <rdfs : domain rdf : resource="#Aside"/>
</owl : DatatypeProperty>
<Body rdf : resource="#htmlExample"/>
    <hasAside>
        <rdf : Bag rdf : ID="asideExample">
        <rdf : li rdf : datatype="http : //www.w3.org/2001/XML
                Schema #string">List1</rdf : li>
        <rdf : li rdf : datatype="http : //www.w3.org/2001/XML
                Schema #string">List2</rdf : li>
        <rdf : li rdf : datatype="http : //www.w3.org/2001/XML
                Schema #string">List3</rdf : li>
        <rdf : li rdf : datatype="http : //www.w3.org/2001/XML
                Schema #string">List4</rdf : li>
        <rdf : type rdf : resource="#Aside"/>
      </rdf : Bag>
   </hasAside>
</Body>
```

〈Figure 9〉 Aside Class

3.2.4 Article & Section

article은 웹에서 실제로 기술하고자 하는 텍스트 부분이라고 볼 수 있고 section은 그 콘텐츠들을 실제로 그룹으로 묶음으로써 절을 나누어서 문단을 표현하고 article을 기술할 수 있다. 여러 개의 article은 여러 개의 section으로 구분할 수 있고 또한 한 article 내에서도 문단의 구분에 따라서 section으로 묶어서 처리할 수 있기 때문에 실제로 article 클래스가 section에 포함될 수 있고 또한 section도 article에 포함될 수 있다. 따라서 body 요소 안에 기술된 section과 article 요소의 특징은 section은 body와 article 두 요소 다 가질 수 있고, article은 body와 section을 가질 수 있게 된다. 또한 section의 특징으로서 section 내부에서 여러 해더들이 사용될 수 있다.

이러한 특징을 표현하기 위해 section 클래스는 ObjectProperty인 #hasArticle이라는 프로퍼티를 선언하고, 프로퍼터의 도메인은 body 클래스와 section 클래스의 union으로 정의한다. DatatypeProperty인 #hasSectionValue로 section 내의 텍스트 정보를 기술할 수 있으며 hgroup로 section 내부의 해더 정보들을 묶어서 처리해 줄 수 있다. 따라서 section 클래스는 <Figure 10>과 같이 정의된다.

```
    <owl : Class rdf : ID="Section"/>
    <owl : ObjectProperty rdf : ID="hasArticle">
        <rdfs : domain>
            <owl : Class>
                <owl : unionOf rdf : parseType="Collection">
                <owl : Class rdf : about="#Body"/>
                <owl : Class rdf : about="#Section"/>
                </owl : unionOf>
            </owl : Class>
        </rdfs : domain>
        <rdfs : range rdf : resource="#Article"/>
    </owl : ObjectProperty>
    <owl : DatatypeProperty rdf : ID="hasSectionValue">
        <rdfs : range rdf : resource="http :
//www.w3.org/2001/XMLSchema
#string"/>
        <rdfs : domain rdf : resource="#Section"/>
    </owl : DatatypeProperty>
```

〈Figure 10〉 Section Class

article 요소는 header를 가지고 있어서 독립적인 글의 제목들을 나타낼 수 있고 또한 Hgroup으로 묶어서 표현할 수도 있다. article 내부에서 문단을 section으로 나누어서 표현하여 나뉘어진 각각의 section에 또 article을 정의할

수 있도록 하여야 한다. 이러한 문제를 해결하기 위하여 article 클래스는 DatatypeProperty로 #hasHeader를 선언하여 해더정보를 기입할 수 있도록 하였으며, 해더정보를 묶어서 관리할 수 있도록 하기 위해서 #hasHgroup 프로퍼티도 가지고 있도록 하였다. 마지막으로 article 내부에서 사용될 수 있는 section 문제를 해결하기 위하여 ObjectProperty인 #hasSection을 선언하여 article 내부에서 삽입된 section 정보의 구조적 표현도 가능하도록 하였다. 따라서 article 클래스는 <Figure 11>과 같이 정의된다.

```
<owl : Class rdf : ID="Article"/>
<owl : ObjectProperty rdf : ID="hasSection">
  <rdfs : domain>
      <owl : Class>
        <owl : unionOf rdf : parseType="Collection">
            <owl : Class rdf : about="#Body"/>
             <owl : Class rdf : about="#Article"/>
        </owl : unionOf>
      </owl : Class>
  </rdfs : domain>
  <rdfs : range rdf : resource="#Section"/>
<owl : ObjectProperty rdf : ID="hasHeader">
  <rdfs : domain>
      <owl : Class>
       <owl : unionOf rdf : parseType="Collection">
        <owl : Class rdf : about="#Body"/>
         <owl : Class rdf : about="#Article"/>
        </owl : unionOf>
       </owl : Class>
  </rdfs : domain>
  <rdfs : range rdf : resource="#Header"/>
</owl : ObjectProperty
```

〈Figure 11〉 Article Class

3.2.5 Footer

footer 요소는 문서의 아랫단에 들어가는 요소로서 저작권, 저자, 서비스 제공자 등 정보들을 담을 수 있다. footer에는 보통 텍스트 정보로 저작권을 표시하거나 혹은 이웃 사이트로의 하이퍼링크를 표기하여 다른 사이트로의 용이한 이동을 제공한다. 따라서 이와 같은 문제를 해결하기 위하여 DatatypeProperty인 #hasFooterValue를 선언함으로 footer의 텍스트 정보를 표현하고, 또 footer클래스도 ObjectProperty인 #hasNav를 갖도록 하여 footer에 기입된 다른 사이트로의 Navigation 정보도 표현이 가능하도록 한다. 또한 footer 요소는 body 요소 내부에 기입됨으로 둘 사이와 관계를 ObjectProperty인 #hasFooter로 정의할 수 있다. 따라서 footer 클래스는 <Figure 12>와 같이 정의된다.

```
owl : Class rdf : ID="Footer"/>
<owl : ObjectProperty rdf : ID="hasNav">
  <rdfs : domain>
      <owl : Class>
       <owl : unionOf rdf : parseType="Collection">
          <owl : Class rdf : about="#Body"/>
          <owl : Class rdf : about="#Footer"/>
       </owl : unionOf>
      </owl : Class>
  </rdfs : domain>
  <rdfs : range rdf : resource="#Nav"/>
</owl : ObjectProperty>
<owl : DatatypeProperty rdf : ID="hasFooterValue">
  <rdfs : domain rdf : resource="#Footer"/>
</owl : DatatypeProperty>
```

〈Figure 12〉 Footer Class

3.2.6 Hgroup

hgroup은 제목을 나타내는 요소로써 여러 개의 head line을 하나로 묶어 표현될 수 있다. 이는 타이틀이라는 구조의 시맨틱 요소를 구조적으로 표현할 수 있는 요소다. 이 타이틀의 묶음은 원칙적으로 body 요소 내의 임의의 구조적 요소 내에서 사용 가능하도록 해야 한다. 따라서 이는 ObjectProperty인 #hasHgroup을 선언하고 프로퍼티의 도메인을 body, header, article, section 클래스의 union으로 설정해줌으로 정의할 수 있다.

OWL로 정의한 hgroup 클래스는 <Figure 13>과 같다.

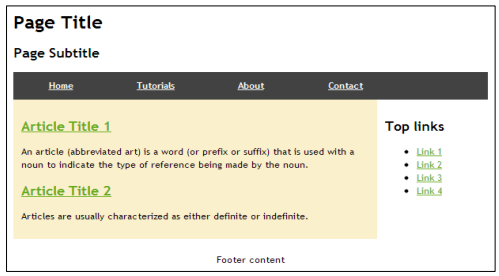
```
<owl : Class rdf : ID="Hgroup"/>
  <owl : ObjectProperty rdf : ID="hasHgroup">
    <rdfs : domain>
      <owl : Class>
        <owl : unionOf rdf : parseType="Collection">
          <owl : Class rdf : about="#Body"/>
          <owl : Class rdf : about="#Header"/>
          <owl : Class rdf : about="#Section"/>
        <owl : Class rdf : about="#Article"/>
        </owl : unionOf>
        </owl : Class>
    </rdfs : domain>
    <rdfs : range rdf : resource="#Hgroup"/>
</owl : ObjectProperty>
```

〈Figure 13〉 Hgroup Class

이렇게 HTML5에 새롭게 추가된 시맨틱 요소들과 각각 매핑된 OWL 모델은 아래의 인스턴스 단계에서 정의된 OWL 모델에 좇아 표현해주는 인스턴스들을 생성해줌으로써 HTML 문서의 온톨로지 표현이 가능해지게 된다.

3.3 HTML5를 통해 OWL 인스턴스 생성

본 절에서는 제 3.2절에서 정의한 OWL 모델에 실제 인스턴스가 어떻게 정의되는지를 보여줌으로써 실제 예제를 바탕으로 인스턴스 정의관계를 보여준다.



〈Figure 14〉 HTML5 Example Page

<Figure 14>는 본 연구에서 사용한 HTML5 예제페이지를 보여준다. 각 부분에 대한 OWL 인스턴스 생성 예제는 아래와 같이 정의된다. 과정은 일대일 매핑 과정으로 수작업으로 완성한다.

3.3.1 Header

header 인스턴스는 header 클래스의 인스턴스로서 문서나 문단의 제목을 의미하며 h요소 혹은 h요소를 하나로 묶어주는 hgroup을 가질 수 있다. <Figure 15>는 실제 웹 페이지에서 보여지는 header 요소이다.

```
<header>
    <hgroup>
      <h1>Page Title</h1>
      <h2>Page Subtitle</h2>
    </hgroup>
    </header>
              ...
<header>
      <h2><a href="#">Article Title 1 </a></h2>
</header>
```

〈Figure 15〉 Header Element in HTML Web Page

<Figure 16>은 <Figure 15>의 요소를 OWL 인스턴스로 매핑한 결과이다.

```
<Header rdf : ID="page_title_header">
  <hasHeaderType rdf : datatype="http : //www.w3.org/2001/XML
      Schema#int">1</hasHeaderType>
          <headerValue rdf : datatype="http : //www.w3.org/2001/XML
          Schema#string">Page Title</headerValue>
          </Header>
          <Header rdf : ID="page_subtitle_header">
            <hasHeaderType rdf : datatype="http : //www.w3.org/
              2001/XMLSchema#int">2</hasHeaderType>
            <headerValue rdf : datatype="http : //www.w3.org/
              2001/XMLSchema#string">Page Subtitle
              </headerValue>
          </Header>
          <Header rdf : ID="hgroupExample">
```

```
<hasHgroup>
  <Hgroup rdf : ID="hgroup_1">
    <hasHeader>
      <Header rdf : ID="page_title_header">
      <hasHeaderType rdf : datatype="http : //www.
        w3.org/2001/XMLSchema#int">1
        </hasHeaderType>
        <headerValue rdf : datatype="http : // www.
      w3.org/2001/XMLSchema#string">
        Page Title
      </headerValue>
        </Header>
    </hasHeader>
    <hasHeader>
      <Header rdf : ID="page_subtitle_header">
        <hasHeaderType rdf : datatype="http : //ww
          w.w3.org/2001/XMLSchema#int">2
        </hasHeaderType>
        <headerValue rdf : datatype="http : //www.w
      3.org/2001/XMLSchema#string">Page Subtitle
      </headerValue>
      </Header>
    </hasHeader>
  </Hgroup>
</hasHgroup>
</Header>
```

〈Figure 16〉 Header Instance Mapping Result

3.3.2 Nav

nav 인스턴스는 body 요소에서 해당 웹 페이지의 중요한 링크를 네비게이션 형식으로 제공해주는 요소이다. 따라서 nav 클래스의 인스턴스도 rdf : Bag 타입으로 정의된다. <Figure 17>은 실제 웹 페이지에서 보여주는 nav 요소이다.

```
<nav>
    <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">Tutorials</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Contact</a></li>
    </ul>
</nav>
```

〈Figure 17〉 Nav Element in HTML Web Page

<Figure 18>은 <Figure 17>의 nav 요소를 OWL 인스턴스 매핑한 결과이다.

```
<Body rdf : ID="bodyExample">
        <hasNav>
          <rdf : Bag rdf : ID="navigationExmaple">
            <rdf : li rdf : datatype="http : //www.w3.org/2001/XML
            Schema #string">About</rdf : li>
            <rdf : li rdf : datatype="http : //www.w3.org/2001/XML
            Schema #string">Contact</rdf : li>
<rdf : li rdf : datatype="http : //www.w3.org/2001/XML
            Schema #string">Tutorial</rdf : li>
<rdf : li rdf : datatype="http : //www.w3.org/2001/XML
            Schema #string">Home</rdf : li>
            <rdf : type rdf : resource="#Nav"/>
          </rdf : Bag>
        </hasNav>
</Body>
```

〈Figure 18〉 Nav Instance Mapping Result

3.3.3 Aside

aside인스턴스는 본문과 관련이 없는 내용을 기술하는 사이드바처럼 사용하는 요소이다. 본 예제에서는 본 웹 페이지에서 다른 페이지로의 하이퍼링크를 aside 요소에서 기술하였다.

<Figure 19>는 웹 페이지에서 보여지는 aside 요소 이다. 보면, aside 내부에 h 요소로 타이틀을 표시하고 ul 요소로 하이퍼링크를 표시하였다.

```
<aside>
        <h2>Top links</h2>
        <ul>
          <li><a ref="external" href="#">Link 1</a></li>
          <li><a ref="external" href="#">Link 2</a></li>
          <li><a ref="external" href="#">Link 3</a></li>
          <li><a ref="external" href="#">Link 4</a></li>
      </ul>
</aside>
```

〈Figure 19〉 Aside Element in HTML Web Page

<Figure 20>은 <Figure 19>의 aside 요소를 제 3.2절에서 정의한 스키마에 근거하여 인스턴스 매핑한 결과이다.

```
<Body rdf : ID="bodyExample">
        <hasAside>
            <rdf : Bag rdf : ID="asideExample">
              <rdf : li rdf : datatype="http : //www.w3.org/2001/XML
               Schema #string">Link1</rdf : li>
              <rdf : li rdf : datatype="http : //www.w3.org/2001/XML
               Schema #string">Link2</rdf : li>
 <rdf : li rdf : datatype="http : //www.w3.org/2001/XML
               Schema #string">Link3</rdf : li>
 <rdf : li rdf : datatype="http : //www.w3.org/2001/XML
               Schema #string">Link4</rdf : li>
              <hasHeader>
                <Header rdf : ID="asideHeader_1">
                  <hasHeaderType rdf : datatype="http : //www.w3.org/20
                   01/XMLSchema#int">2</hasHeaderType>
                  <headerValue rdf : datatype="http : //www.w3.org/20
                   01/XMLSchema#string">Top Links</headerValue>
                </Header>
              </hasHeader>
             <rdf : type rdf : resource="#Aside"/>
            </rdf : Bag>
          </hasAside>
 </Body
```

〈Figure 20〉 Aside Instance Mapping Result

3.3.4 Article & Section

제 3.2절의 스키마 정의에서 언급한 것처럼, 실제 웹 페이지의 텍스트를 기술하는 article은 section으로 문단을 나눌 수 있고 또한 section으로 article 내의 여러 문단을 나타낼 수도 있다. 즉 article과 section은 서로 포함하는 관계를 가지고 있다. <Figure 21>은 section 내에 article이 포함되어 있고 article 내에 또 하나의 inner article이 포함된 경우의 예를 나타낸다.

<Figure 22>는 제 3.2절에서의 스키마 매핑 방법에 따라 <Figure 21>의 section & article 요소를 매핑한 결과이다.

```
<section id="articles">
  <article>
       <header>
            <h2><a href="#">Article Title 1 </a></h2>
       </header>
            <section>An article (abbreviated art) is a word
                              (or prefix or suffix) that is used
                          with a noun to indicate the type of
                          reference being made by the noun.
           </section>
  </article>
  <article>
            <header>
            <h2><a href="#">Article Title 2</a></h2>
            </header>
             <section>Articles are usually characterized as
               either definite or indefinite.
            </section>
  </article>
</section>
```

〈Figure 21〉 Article & Section Element in HTML Web Page

```
<Body rdf : ID="bodyExample">
<hasSection>
            <Section rdf : ID="sectionExample_ID_articles">
              <hasArticle>
                <Article rdf : ID="articleExample2">
                  <hasSection>
                    <Section rdf : ID="sectionExample_belongToArticle2">
                      <hasSectionValue rdf : datatype=
                        "http : //www.w3.org/2001/XMLSchema#string">
                          Articles are usually characterized as
                          either definite or indefinite. </hasSectionValue>
                    </Section>
                  </hasSection>
                 <hasHeader rdf : resource="#article_header_2"/>
                </Article>
              </hasArticle>
             <hasArticle>
             <Article rdf : ID="articleExample1">
                 <hasSection>
                   <Section rdf : ID="sectionExample_ belongToArticle1">
                     <hasSectionValue rdf : datatype=
                       "http : //www.w3.org/2001/XMLSchema#string">
                       An article (abbreviated art) is a word
                           (or prefix or suffix) that is used with
                           a noun to indicate the type of reference
                           being made by the noun.</hasSectionValue>
                    </Section>
                   </hasSection>
                  <hasHeader        rdf : resource="#article_head-
er_1"/>
                 </Article>
               </hasArticle>
             </Section>
           </hasSection>
             </Body>
```

〈Figure 22〉 Article & Sectioninstance Mapping Result

3.3.5 Footer

Footer 인스턴스는 저작권, 저자 등 웹 페이지 관련 메타데이터를 기술할 수 있는 요소이다. <Figure 23>은 웹 페이지에서 표현한 footer 요소이다. 제 3.2절에서 스카마를 정의할 때에 언급 하였듯이 footer 요소에서는 텍스트 정보를 담을 수 있을 뿐 아니라, 다른 사이트로의 하이퍼링크도 기술할 수 있다. 본 인스턴스 매핑 예제에서는 foot요소에 텍스트 형식으로 기술된 것을 대상으로 하였다.

```
<footer>
    Footer content
</footer>
```

〈Figure 23〉 Footer Element in HTML Web Page

<Figure 24>는 <Figure 23>의 footer 요소를 인스턴스 매핑한 결과이다.

```
<Body rdf : ID="bodyExample">
    <hasFooter>
        <Footer rdf : ID="footerExample">
            <hasFooterValue rdf : datatype="http : //www.w3.org
            /2001/XMLSchema#string">Footer Content
            </hasFooterValue>
        </Footer>
    </hasFooter>
</Body>
```

〈Figure 24〉 Footer Instance Mapping Result

3.3.6 Hgroup

section이나 aside 등 다른 요소들은 전체 웹 페이지를 대상을 구조적인 방식으로 표현하려고 하였다면, hgroup 요소는 웹 페이지에 기술된 텍스트에 대한 구조적 표현이라 볼 수 있다. 즉 문서 텍스트의 타이틀에 대하여 묶어서 표현함으로 텍스트의 타이틀이라는 구조를 표현할 수 있다. <Figure 25>는 웹 페이지에서의 hgroup 요소이다.

```
<hgroup>
<h1>Page Title</h1>
<h2>Page Subtitle</h2>
</hgroup>
```

〈Figure 25〉 Footer Element in HTML Web Page

<Figure 26>은 hgroup 스키마 정의에 따라 웹 페이지 hgroup 요소를 인스턴스 매핑한 결과이다.

```
<Hgroup rdf : ID="hgroup_1">
    <hasHeader>
        <Header rdf : ID="page_title_header">
            <hasHeaderType rdf : datatype="http : //www.w3.org
/2001/XMLSchema#int">1</hasHeaderType>
<headerValue rdf : datatype="http : //www.w3.org/20
01/XMLSchema#string">Page Title</headerValue>
        </Header>
    </hasHeader>
    <hasHeader>
        <Header rdf : ID="page_subtitle_header">
            <hasHeaderType rdf : datatype="http : //www.w3.org
/2001/XMLSchema#int">2</hasHeaderType>
<headerValue rdf : datatype="http : //www.w3.org/20
01/XMLSchema#string">Page Subtitle</headerValue>
        </Header>
    </hasHeader>
</Hgroup>
</hasHgroup
```

〈Figure 26〉 Hgroup Instance Mapping Result

4. 결론 및 향후 연구 방향

본 연구에서는 HTML5에서 새로 추가된 시맨틱 요소들을 대상으로 웹 페이지의 온톨로지

화 방법론을 제안하였다. 시맨틱 웹의 발전으로 말미암아 웹의 표현언어도 점차 더 풍부해질 것이다. 어떻게 하면 웹에 있는 데이터들을 더 의미있게 표현할 것인가 하는 노력들이 많이 진행되어 왔으며 그 결과로 HTML5 및 그에 종속된 마이크로데이터, RDFa와 같은 기술들이 나타나고 표준화가 진행되고 있다. 그러므로 본 논문에서는 HTML5에 새로 추가된 구조적 시맨틱 요소들을 구조적인 각도에서 온톨로지화 하는 방법론을 제안함으로써 앞으로 웹의 표준이 될 HTML5문서의 온톨로지화 작업에 대한 밑거름이 되는 데 의미가 있다고 본다. 또한, HTML5로 작성된 문서를 온톨로지로 변환하는 방법을 소개함으로써 이를 통해 HTML5 문서의 공유 및 재사용, 상호운용성이 가능하게 될 것이라 기대한다.

본 연구에서는 Heading과 Sectioning에 포함된 요소들을 바탕으로 온톨로지화 작업을 수행하였다. 따라서 향후 본 연구에서는 HTML5에 포함된 모든 정보들을 온톨로지로 표현할 수 있게 하는 연구를 진행하고자 한다.

References

[1] Brian, E., Chapman, M. W., Claudiu F., and Patrick, R., “Annio : A Web-based Tool for Annotating Medical Images with Ontologies,” IEEE Second Conference on Healthcare Informatics, Imaging and Systems Biology, 2012.

[2] Hwangbo, H. and Lee, H. C., “Reusing of Information Constructed in HTML Documents : A Conversion of HTML into OWL,” International Conference on Control, Automation and Systems Proceedings, Vol. 7, pp. 871-875, 2008.

[3] http://ko.wikipedia.org/wiki/Ontology.

[4] http://alistapart.com/article/previewofHTML5.

[5] http://www.whatwg.org/specs/web-apps/current-work/multipage/microdata.HTML.

[6] Kim, K. Y. and Lee, W. Y., “RDF Metadata Editor for Semantic Web Environment,” Korean Institute of Industrial Engineers Autumn Simposium 2004 Proceedings, Vol. 15, pp. 685-688, 2004.

[7] Kwak, H. S., Kim, S. K., Kim, Y. G., and Ahn, K. H., “A Conversion System of HTML Document into OWL Ontology Language,” Korea Information Processing Society Conference, pp. 539-542, 2004.

[8] Lee, M. K., Ha, Y., and Kim, Y. S., “The RDF Metadata Generation System for WebSite Management,” Journal of KIISE : Software and Applications, Vol. 28, No. 4, 2001.

[9] Lee, M. H., Cho, H. K., Cho, H. S., Cho, S. H., Jang, C. B., and Choi, E. I., “OWL Authoring System for building web Ontology,” The Journal of Society for e-Business Studies, Vol. 10, No. 3, 2005.

[10] www.w3c.org.

[11] Yun, S. H., Shin, H. S., Jeong, C. M., Kyoung, J., and Kwon, C., “HTML5 Guide,” pp.10, Web Standard Korea, 2010.

저 자 소 개

손태모 (E-mail : johannessohn@yonsei.ac.kr)
현재 연세대학교 정보산업공학과 석사과정
관심분야 시맨틱 웹과 의미기반 검색 및 데이터마이닝 등

윤이연 (E-mail : pryiyeon@yonsei.ac.kr)
현재 연세대학교 정보산업공학과 석박통합과정
관심분야 시맨틱 웹과 데이터마이닝, 인공지능, 통계기법 등

김우주 (E-mail : wkim@yonsei.ac.kr)
1987년 연세대학교 BBA과정 (학사)
1994년 KAIST 경영과학 (박사)
현재 연세대학교 정보산업공학과 교수
관심분야 시맨틱 웹, 시맨틱 웹 환경의 의사결정지원 시스템,
시맨틱 웹 마이닝, 지식관리 및 인공지능 웹 서비스 등