주 제

차세대 웹 환경에서의 다국어 식별자 기술 동향

안양대학교 정의현

차례

I. 서 론

1.1. 차세대 웹의 출현

웹(Web)의 출현 이후로 웹이 현대 사회에 기여한 학술적/산업적 성과는 매우 주목할 만한 것이다. 웹은 단순히 기술을 넘어서 산업계를 좌지우지하는 거대한 비즈니스로 각광받고 있으며, 대중에게 가장 밀접한 정보의 원천 역할을 하고 있어, 웹이 없는 현대 사회는 상상할 수 조차 없다. 이렇듯 웹의 발전은 단순한 기술적 진보를 넘어서 인터넷 산업의 가장 중요한 기술적 요건이 되고 있다. 그러나 현재의 웹은 HTML(Hyper Text Markup Language)을 브라우저로 보여주는 표현(presentation) 중심의 웹이기 때문에 씬(thin) 클라이언트로의 역할이라는 한계점을 갖고 있다. 이러한 웹 기술의 한계는 학계와 산업계에 새로운 방향에 대한 모색을 하도록 하였으며, 그 대답이 각각 시맨틱 웹(Semantic Web)[1]과 웹 2.0[2]이다. 시맨틱 웹은 기계가 이해할 수 있고, 기계간의 자동화된 데이터 소통이 가능한 웹 구조에 대한 기술적 흐름이며 학계와 W3C(World Wide Web Consortium)가 중심이 되어 연구를 진행하고 있다. 이에 비해 웹 2.0은 산업계 내부에서 사용자에게 새로운 서비스의 경험을 제공하고, 새로운 비즈니스 기회를 얻기 위한 목적으로 제시되었다[3]. 비록 웹 2.0이 시맨틱 웹에 비해서 기술의 심도에 대한 비판이 있는 것은 사실이나 웹 비즈니스에서는 실제 활용이 가능한 실용적(practical) 기술로 인정받고 있는 상황이다. 이 두 가지 주제는 차세대 웹에 대한 전혀 다른 방향을 의미하는 것은 아니며 장기적으로는 협력적으로 융합되어 차세대 웹의 기술적 기반을 이룰 것으로 판단된다. 웹 2.0의 기술적 요소인 집단지성

(Folksnomy), 개방형 API(Open API), 리치 클라이언트(rich client) 기술은 웹의 표면(surface)을 담당하게 될 것이며, 시맨틱 웹은 해당 서비스들의 자동화된 데이터 결합을 가능하게 하는 웹의 중심(core) 역할을 하게 될 것으로 예측된다.

웹 2.0과 시맨틱 웹에서 공통적으로 인터넷 식별자(URI: Uniform Resource Identifier)[4]는 매우 중요한 역할을 하고 있다. 웹 2.0에서는 짧고 이해하기 쉬우며, 불변하는 인터넷 식별자를 만들 것을 주장하고 있다[2]. 이는 기존 웹과 달리 웹 2.0에서는 페이지의 식별자가 더욱 중요한 의미를 갖기 때문인데, 웹 2.0에서는 인터넷 식별자가 많은 페이지들을 서로 연계하고 가치를 부여하는 링크(link)의 기본이 되기 때문이다. 따라서 인터넷 식별자는 쉽게 붙여 넣기 쉽도록 짧으면서 직관적이어야 한다. 또한 이 식별자가 시스템의 구성 변화에 상관없이 영속적이어야 한다. 이렇게 되어야 RSS(Really Simple Syndication)나 퍼머링크(permalink), 트랙백(Trackback)으로 연결된 블로그(Blog)들의 결합이 망가지는 것을 막을 수 있다. 이렇게 직관적이면서 이해하기 쉬운 인터넷 식별자를 갖기 위해서는 모국어를 사용하는 것이 바람직하다. 그러나 현재의 인터넷 식별자 표준인 URI는 US-ASCII 만을 사용하도록 규정되어 있는 한계점이 있다[4]. 이러한 요구 사항은 시맨틱 웹에서도 마찬가지이다. 시맨틱 웹은 기본적으로 리소스(resource)와 술어(predicate)의 네트워크에 의존하고 있다[5]. 따라서 시맨틱 웹에서는 리소스의 구별뿐 아니라 술어의 지정에 대해서도 인터넷 식별자는 중요하게 사용되고 있으며, 기계들의 추론에서도 이러한 식별자의 사용은 필수적이라 할 수 있다. 이러한 관점에서 모호성(ambiguity)이 증대되는 영문 식별자를 리소스나 술어를 표시하기 위해 사용하는 것은 의미를 정확히 서술하는 것이 주요한 목적인 시맨틱 웹의 기본 개념에도 적합하지 않다. 이러한 차세대 웹의 기술적 요구사항을 고려할 때, 차세대 웹에서는 모국어를 이용해서 리소스를 식별할 수 있는 다국어 식별자(Internationalized Resource Identifier)의 도입이 필수적이라 하겠다.

1.2. 인터넷 식별자의 국제화 요구 및 국내 현황

웹(Web)의 주요한 목적 중의 하나는 어떤 주체라도 다른 주체들과 정보를 쉽게 공유할 수 있도록 구성된 전역적인(global) 정보공간을 구축하는 것이다. 이러한 목적을 성취하기 위해서는 무엇보다도 정보공간 내의 자원(resource)들을 통일된 방법으로 식별할 수 있는 시스템(identification system)이 요구된다[6]. W3C에서는 이러한 목적을 위하여 자원의 통일된 식별 체계로 인터넷 식별자(URI: Uniform Resource Identifier)[4]를 제시하였다. 인터넷 식별자는 인터넷 상의 텍스트, 비디오, 음향, 이미지 및 기타 서비스의 식별을 위해 사용되며, URI의 가장 대표적인 형식은 웹 페이지 주소로, 모든 자원 접근 메커니즘, 자원 소속 컴퓨터, 자원 명칭 등이 이 형식으로 표현된다.

인터넷 식별자가 비록 성공적인 식별 체계로서 널리 사용되었지만, 웹 환경의 변화에 따라서 기존 인터넷 식별자 표준으로는 지원될 수 없는 몇 가지 요구사항이 대두되었다. 이 중에서 가장 커다란 요구 사항은 "국제화(Internatialization)"에 관한 요구사항이다. URI는 단순히 리소스의 위치를 나타낼 뿐 아니라, 의미 있는 단어로 쉽게 이해할 수 있도록 하기 위한 의도를 갖고 구축되어 있다. 즉 URI는 그 개념자체에, 어떤 대상을 가리키는 식별자(identifier)가 단지 의미 없는 숫자/문자의 조합을 가정하기보단, 기억하기 쉽고, 해석하기 쉽고, 변환하기 쉽고, 생성하기 쉽고, 추측하기 쉬운 표식으로 그 표식을 사용할

인간/기계 모두에게 원활한 표현 방법의 표준을 정하는 것이라 할 수 있다. 이를 위해서 URI를 구성하는 단어들은 자연어/모국어(natural language)를 나타내는 문자들로 구성되는 것이 바람직하다. 그러나 기존 URI는 US-ASCII 문자 집합으로만 URI를 구성할 수 있도록 구성되어 있어, 영어권 외의 다른 언어권에서는 URI를 구성하기 위해 자신의 모국어에 대응되는 영문으로 변환하여 URI를 구성해야만 했다. 이러한 영문 변환을 이용한 URI 구성은 모호성을 가중시키게 되며, 사용자의 불편을 야기하게 된다. 또한 한글 도메인이나 2단계 kr 도메인 도입 등 도메인 사용 환경 변화 때문에 URI에 대한 국제화 요구는 단순히 불편함의 범위를 넘어서 국내 정보통신 기술 발전을 위해 반드시 해결되어야 하는 주요 과제로 부각되고 있다. 이를 해결하기 위해 유니코드(Unicode) [7]를 적용한 확장 URI 표준인 IRI(Internationalized Resource Identifier; 이하 다국어 식별자) [8]가 W3C와 IETF에 의해서 2005년에 표준으로 제시되었다. 그러나 현재 이 표준안을 한글에 적용하기 위한 체계적인 표준화 연구는 기초적인 상태이며, 국내에는 EUC-KR과 UTF-8에 기반한 한글 인코딩(encoding) 방식이 혼재되어 사용되고 있어 다국어 식별자의 실질적인 적용에는 어려움이 예상된다. 그러나 자원의 식별을 제공하는 식별 체계는 여러 정보 서비스의 도입과 상호운용성을 보장하는 기초적인 기술이다. 따라서 웹 기술과 정보통신 기술이 활발한 국내 상황에는 "국제화"가 고려된 URI 표준안의 IT 기술에의 적용은 중요한 주제라 할 것이다.

본 기고에서는 다국어 식별자의 구조와 이에 따른 차세대 웹 환경에서의 적용에 관해서 논의하고자 한다. 2장에서는 다국어 식별자의 적용에 필수적인 유니코드에 관한 기술적인 개념과 다국어 식별자의 개요에 관해 설명한다. 3장에서는 네트워크 전송을 위해 다국어 식별자를 어떻게 인터넷 식별자로 변환하는지에 관해서 설명한다. 4장에서는 국내의 웹 환경에서의 적용 방안에 관해서 살피고, 5장에서 결론을 맺는다.

II. 유니코드와 다국어 식별자

다국어 식별자에 대한 기술적 개념을 이해하기 위해서는 문자 인코딩(character encoding)에 대한 개념 이해가 필수적이다. 따라서 이 장에서는 유니코드에 대한 기술적 개념을 간단히 살펴보도록 한다.

2.1. 유니 코드의 문자 인코딩

문자도 컴퓨터 내에서는 숫자로 코딩(coding)되어 저장되어야 한다. 코딩의 일반적인 방법은 0부터 255까지의 숫자를 사용하는 것인데, 이것은 데이터 저장과 전송의 기본 단위인 바이트(byte) 혹은 옥텟(octet)에 적합하기 때문이다. 문자를 숫자에 코딩하기 위해서는 각 문자 별로 대응되는 숫자를 정의한 대응표(mapping table), 즉 문자 코드(Character code)가 정의되어야 하며, 대표적인 문자코드로는 ASCII나 EBCDIC 등의 코드가 있다. ASCII의 문자 영역은 0~127까지의 값으로 충분히 대응되기 때문에 128~255까지의 상위 영역은 임의의 문자로 채워질 수 있다. 이 상위 영역을 이용하여 여러 언어들이 자신의 고유한 문자를 대응시킨 문자 코드를 제시하였다. 예를 들어 Latin-1 문자코드는 프랑스어나 독일어의 확장 문자(ö 나 é 등)를 상위 영역에 배치하여 서유럽 언어들을 하나의 문자 코드에서 지원할 수 있었다. 그러나 이러한 접근 방안은 여러 언어를 하나의 문서에서 혼용해서 사용해야 하는 경우, 특히 그 언어들이 다른 문자코드를 사용해야 하는 경우가 문제를 야기한다. 예를 들어 영어와 불어를 같이 쓰거나 영어

와 그리스어를 같이 쓰는 경우는 아무런 문제가 되지 않는다. 그러나 불어와 그리스어를 같이 쓰는 경우에는 128~255의 상위 영역이 겹치는 문제가 발생하게 된다. 또한 동일한 문자 코드를 사용하는 언어권 내에서 디지털 문서가 교환되는 경우에는 아무런 문제가 없으나, 다른 문자 코드로 표현되는 언어권 사이에서 디지털 문서가 교환되는 경우에는 호환성에 문제를 발생시킨다. 예를 들어 Latin-1 문자 코드를 이용한 resume 라는 글자를 그리스에서 자국어의 ISO-8859-7 문자코드를 이용해서 읽으면 rεsumε로 나타나게 된다. 이는 문자 코드의 같은 데이터 영역을 각 언어권의 다른 문자가 대응되기 때문이다. 또 한가지의 큰 문제점은 한글, 중국어 등의 동아시아권의 문자들은 애초부터 1 바이트의 영역을 넘는 문자 크기를 갖고 있기 때문에 이러한 접근 방안 자체가 유효한 방법이 아니다.

정보화가 급속히 진행되고, 세계적으로 정보 교류가 활발한 시점에서 상기의 문제점에 대한 해결책으로 제시된 것이 바로 유니코드(Unicode)이다. 유니코드는 세계 모든 언어들의 모든 문자들에게 유일(unique)하고 불변(unchanged)한 숫자값을 부여하는 문자코드이다. 이 번호는 문서에서 사용되는 언어에 종속되지 않고, 문자를 나타내는 폰트에 종속되지 않고, 소프트웨어, 운영체제, 디바이스에 종속되지 않는 고유 번호이다. 이것은 고유할 뿐 아니라 불변의 값을 가진다. 또한 가능한 숫자의 범위는 현존하는 언어와 미래의 언어의 필요를 모두 포함할 수 있을 정도로 크게 정의되어 있다. 유니코드 표준 용어로 코딩 스페이스(Coding Space)는 문자들을 지정하기 위한 숫자의 범위를 의미한다. 8비트 인코딩에서는 코딩 스페이스가 0부터 255까지이다. 유니코드에서의 코딩 스페이스는 0부터 10FFFF까지 이며, 10진수로는 1,114,111이다. 유니코드에서의 각 위치는 코드 포인트(code point) 혹은 코드 포지션(code position)라 하며, 코딩 스페이스에서의 값을 의미한다.

유니코드가 실제로 물리적으로 저장되거나 네트워크로 전송되기 위해서는 특정 옥텟 스트림으로 변환되어야 한다. 이를 인코딩(encoding)이라 하며, 문자를 나타내는 코드 넘버를 코드 단위(unit)로 매핑(mapping)하는 것을 의미한다. 코드 단위란 옥텟(octet), 더블 옥텟(double octet), 쿼드러플 옥텟이다(quadruple octet)으로 한 묶음 단위의 바이트 배열을 의미한다. 일반적으로 유니코드의 가장 쉬운 인코딩 방식은 각 코드 넘버를 쿼드러플 옥텟으로 매핑시키는 것이다. 이러한 인코딩을 UTF-32라고 한다. 그러나 현재의 유니코드가 BMP(Basic Multilingual Plane)의 16비트의 값을 주로 이용하기 때문에 실용적인 면에서는 너무 비효율적이다. 이런 이유 때문에 BMP 영역의 16비트 값을 직접 더블 옥텟으로 매핑할 수 있는 UTF-16이 제안되었다. UTF-16은 직관적인 방식이기는 하지만, ASCII 코드와의 호환성은 전혀 없는 인코딩 방식이다. 이 때문에 기존 문서들과 호환되기 어려운 단점이 있는데, 이를 해결하기 위한 방식이 바로 UTF-8[9]이다. UTF-8은 문자 코드에 따라서 가변적인 코드 단위가 사용된다. U+0000부터 U+007F까지의 코드 영역은 ASCII와 동일한데 이 영역은 변환 없이 바로 1 바이트로 변환된다. 따라서 UTF-8은 기존의 영문자와 호환성이 좋은 것이 특징이다. 이러한 점은 반대로 영문자 외의 문자들은 모두 두 개 이상의 옥텟으로 대응되어야 한다는 단점을 내포한다. 다음 표는 문자열 "pâté"를 여러 인코딩으로 표현한 것이다. (UTF-16LE는 리틀 엔디언(Little Endian)을 의미한다.) 표에서 알 수 있듯이 동일한 유니코드의 문자 코드라도 인코딩 방식에 따라서 다른 옥텟 스트림으로 변환됨을 알 수 있다.

〈표 1〉 유니코드의 인코딩

유니코드	U+0070, U+00E2, U+0074, U+00E9
UTF-8	70 C3 A2 74 C3 A9
UTF-16	0070 00E2 0074 00E9
UTF-16LE	7000 E200 7400 E900
UTF-32	00000070 000000E2 00000074 000000E9
Latin-1	70 E2 74 E9

2.2. 다국어 식별자 표준안

인터넷 식별자의 문자들은 종종 일상에서 사용되는 단어를 나타내기 위해서 사용된다. 이러한 사용 방법은 많은 장점을 갖고 있는데, 먼저 자연어를 이용한 이런 인터넷 식별자는 기억하기 쉽고, 해석하기 쉽고, 옮기기 쉽고, 생성하기 쉽고, 추측하기 쉽다. 그러나 기존의 인터넷 식별자는 US-ASCII 문자집합의 하위 집합에서 선택된 문자들만 사용할 수 있기 때문에 영어를 제외한 다른 언어권에서는 자신의 문자를 알파벳 문자로 바꿔서 써야만 한다. 이런 번역과정이 인터넷 식별자 사용에 종종 이용되지만, 이러한 접근 방안은 인터넷 식별자의 사용에 있어서 모호성을 가중시킨다. 예를 들어, "http://www.example.org/친구_얼굴" 이라는 인터넷 식별자가 친구 얼굴에 대한 인터넷 자원을 나타내는 것이라면, 이 인터넷 식별자가 무엇을 의미하는지는 한글을 읽을 수 있는 대한민국 사람이라면 누구나 알 수 있다. 그러나 이것을 기본적인 인터넷 식별자 표준에 맞추기 위해서는 "http://www.example.org/friend_face"나 아예 의미 없는 "http://www.example.org/aaa_111"로 변환하여 사용해야 한다. 그러나 이런 변환은 컨텐츠 제공자마다 다르게 만들 수 있기 때문에 일관적이고 통일적인 식별자 역할을 하기 어렵다.

현재 다국어 문자를 지원하기 위한 하부구조는 운영체제와 응용 소프트웨어에 널리 채택되고 있다. 이른바 국제화(Internationalization)와 지역화(Localization)를 통하여 다양한 종류의 문자와 언어를 동시에 처리할 수 있는 소프트웨어가 점차적으로 증가하고 있는 것이다. 또한 프로토콜 등도 넓은 범위의 문자들을 수용할 수 있도록 개선되는 중이다. 따라서 리소스를 나타내는 인터넷 식별자도 이러한 국제화와 지역화를 지원할 수 있도록 확장됨이 요구되며, 이의 결과가 바로 다국어 식별자이다. 다국어 식별자는 표 1과 같이 다양한 문자집합을 인터넷 식별자에 사용할 수 있는 특징을 갖고 있다.

〈표 2〉 다국어 식별자의 예

http://www.example.org/Dürst.html
http://www.example.org/한글처리.html
http://www.exampe.org/かお.html

다국어 식별자 표준에 의하면 모든 다국어 식별자는 반드시 유니코드를 기반으로 하도록 되어 있다. 또한 다국어 식별자는 유니코드 기반의 문자집합을 사용하는 프로토콜, 포맷, 소프트웨어 컴포넌트에서 인터넷 식별자를 대체할 목적으로 제안되었다. 하지만 이 프로토콜들과 컴포넌트들은 식별자가 리소스를 지칭하는 목적으로만 사용하는 한, 굳이 인터넷 식별자로 변환을 할 필요가 없다. 그러나 만일 식별자를 리소스 검색(retrieval)에 사용하려 한다면, 기존 인터넷 표준과 호환되는 인터넷 식별자를 결정하는 것이 필요하다. 이는 현재 HTTP 프로토콜[10]은 인터넷 식별자만을 지원하도록 되어 있기 때문이다. 따라서 브라우저와 같은 사용자 인터페이스 영역에서는 다국어 식별자로 표시한다 하더라도, 실제로 인터넷 상에서 리소스를 검색하는데 있어서는 인터넷 식별자로의 변환이 중요한 문제이다. 인터넷 식별자로의 변환은 필연적으로 문자 인코딩과 관련이 되어 있는데, 앞 절에서 살펴본 바와 같이 다국어 식별자의 표시 자체는 유니코드로 표시할 수 있지만, 유니코드의 인코딩을 어떤 것으로 하느냐에 따라서 유니코드로 표시된 식별자가 다르게 해석될 수 있기 때문이다.

예를 들어 UTF-8으로 인코딩된 데이터가 실수로 ISO-8859-1로 인코딩 된것으로 해석된다면 1개의 문자가 2개의 문자로 바뀌어 출력되게 된다. 예를 들어 "Here is my resume"라는 문장을 UTF-8으로 인코딩하여 보냈는데, 이를 ISO-8859-1 인코딩을 적용하여 화면에 출력하면 "Here is my rÃ©sumÃ©"로 출력된다. 따라서 표준적인 방식의 인터넷 식별자와 다국어 식별자의 상호 변환에 대한 내용은 표준에서 매우 중요한 주제이며, 다음 장에서 이 내용을 살펴보도록 한다.

III. 다국어 식별자의 인터넷 식별자 변환

3.1. 변환 방식

다국어 식별자를 인터넷 식별자로 변환하기 위해서는 두 개의 과정을 거쳐야 한다. 첫 번째 과정은 유니코드 문자열을 생성하는 과정이고, 두 번째 과정은 네트워크로 실제 전송이 가능한 형태로 바꾸는 과정이다.

◎ 과정 1

원래의 다국어 식별자로부터 유니코드 문자열을 생성한다. 이 과정은 입력의 형식에 따라 세가지 변환 형태를 갖는다. 첫째는 다국어 식별자가 문서에 써 있거나 어떤 종류의 문자 인코딩과도 상관없이 표시되어 있는 경우이다. 이러한 경우에는 다국어 식별자를 정규표현식 C(Normalization Form C)에 근거한 UCS(Unicode Character Set) 정규화(normalization)를 이용하여 문자열로 표현한다[11]. 두 번째, 만일 다국어 식별자가 비(非)유니코드 문자 인코딩을 사용한 디지털 형태라면 해당 다국어 식별자를 NFC에 근거한 UCS 정규화를 이용하여 변환한다. 마지막으로 만일 다국어 식별자가 유니코드에 기초한 문자 인코딩(예를 들면 UTF-8 이나 UTF-16)이면 정규화를 하지 않고 그대로 놔둔다.

◎ 과정 2

IRI의 스킴(scheme), 도메인 이름, 구분자를 제외한 문자열 중 유니코드 영역의 각 문자에 대해 아래의 과정을 순서대로 적용한다.

(1) 문자를 UTF-8을 이용하여 하나 또는 그 이상의 옥텟 배열로 변환한다.
(2) 각 옥텟을 %HH로 변환한다. HH는 옥텟값의 16진수 표현이다. 이것은 URI 표준의 퍼센트 인코딩(percent-encoding mechanism) 기법과 동일하다. 표현의 모호함을 줄이기 위해 16진수표현은 반드시 대문자를 사용한다.
(3) 원래의 문자를 결과 문자열로 대체한다.

예를 들어 다음과 같은 URI가 있다고 가정하였을 때, 각 과정에 의해서 변화되는 것은 다음과 같다. 첫 번째 단계에서의 < >는 옥텟을 구분하기 위해서 표시한 것으로 실제로는 URI 변환에서 포함되지 않는다.

- http://www.example.org/한글_domain
  (다국어 식별자의 예)
- http://www.example.org/<ed><95><9c><ea><b8><80>_domain (UTF-8 인코딩 변환)
- http://www.example.org/%ED%95%9C%EA%B8%80_domain (% 인코딩)

다국어 식별자를 수용하는 시스템은 또한 인터넷 식별자에서 허가하지 않은 US-ASCII의 출력 가능한 문자들에 대해서 다룰 수 있지만, 과정 2에서 변환

되지 않는다면 인터넷 식별자로의 변환은 실패한다. 숫자 기호 ("#"), 퍼센트 기호 ("%"), 그리고 각진 괄호 문자 ("[", "]") 들은 위의 목록에 해당하지 않으므로 변환되지 않아야 된다.

3.2. 다국어 도메인의 처리

위에 언급한 다국어 식별자로부터 인터넷 식별자로의 변환은 인터넷 식별자 표준을 완전히 준수하는 인터넷 식별자를 생성한다. 그러나 다국어 식별자의 ireg-name 요소가 도메인 이름에 해당되고, 기존의 퍼센트 인코딩을 통해서 변환할 수 없는 구조(scheme)를 갖고 있다면, 다국어 식별자를 수용한 시스템은 ireg-name 부분에 대해서 다국어 도메인(IDNA) [12]과 같은 구조를 통하여 변환할 수 있다. 이러한 변환은 앞 절의 과정 2 이전에 일어나야 한다. 이때는 레이블(label) 구분자로 U+002E(.)를 이용하여 각각의 점(dot)로 구분된 레이블에 다국어 도메인에서 규정한 ToASCII[12] 동작을 이용하여 변환된 부분으로 다국어 식별자의 ireg-name 요소를 대체한다. 예를 들어 다국어 식별자가

"http://résumé.ecample.org"

인 경우, UTF-8 인코딩을 이용하여

"http://r%C3%A9sum%C3%A9.ecample.org"

로 변환하는 대신에 아래와 같이 변환될 수 있다.

"http://xn--rsum-bpad.ecample.org"

이것은 도메인 이름에서 다국어 도메인 표준을 적용하고자 하는 경우에 ToASCII가 사용하는 punycode[13]를 통해서 변환된 것이다. 이것은 다국어 식별자의 표준 영역에 속하는 것은 아니지만, 다국어 도메인이 국내에 도입되는 경우에 도메인 이름에 해당하는 부분은 따로 변환 작업이 이루어져야 DNS를 경유하여 해당 서버로 전송될 수 있다.

IV. 다국어 식별자의 국내 적용 현황

4.1. 인코딩의 혼재

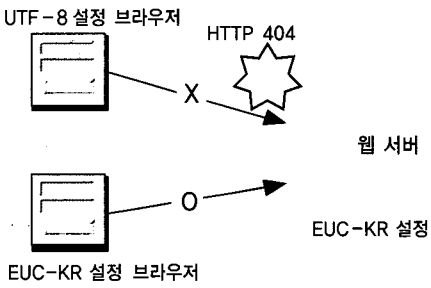
다국어 식별자는 유니코드를 사용하도록 규정되어 있기 때문에 브라우저에서 웹 서버로 전송되기 위해서는 적절한 문자 인코딩을 거쳐야 하며, 표준안에서 제시된 바와 같이 유니코드로 표현된 다국어 식별자는 전송 시에 반드시 UTF-8 인코딩으로 전송되어야 한다. 그러나 국내의 웹에서 사용되는 문자 인코딩 방식은 UTF-8과 EUC-KR 표준으로 나뉘어져 있다. 비록 EUC-KR은 유니코드의 지원이 불가능한 방식이나, 적은 바이트로 한글을 표현할 수 있기 때문에 EUC-KR이 사용되는 경우도 적지 않다. 문제는 이러한 문자 인코딩의 혼용이 다국어 식별자의 사용에 문제가 된다는 점이다.

<표 3>에서 볼 수 있는 것처럼 동일한 다국어 식별자도 문자 인코딩 방식에 따라서 다른 형태의 바이트 스트림으로 서버에 전송되게 된다. 문제는 웹 서버 측에서 사전에 설정된 문자 인코딩으로 전송되지 않은 URI 스트림에 대해서는 해석을 제대로 할 수 없어 인터넷 식별자에 해당하는 실제 리소스 위치를 파악할 수 없다는 점이다.

〈표 3〉 다국어 식별자의 인코딩 표현

방 식	실제 데이터의 형태
다국어 식별자	http://test.org/한글.html
EUC-KR	http://test.org/%C7%D1%B1%DB.html
UTF-8	http://test.org/%ED%95%9C%EA%B8%80.html

예를 들어 (그림 1)과 같이 EUC-KR을 기본 문자 집합으로 설정한 웹 서버에게 웹 브라우저의 인코딩 방식에서 UTF-8 옵션으로 지정하여 인터넷 식별자를 전송하게 되면, URI에 해당하는 리소스에 접근할 수 없기 때문에 웹 서버는 HTTP 404[12] 에러를 발생시키게 된다. 그러나 UTF-8 옵션을 해제한 후에 EUC-KR로 전송하게 되면, 제대로 페이지가 브라우징 되게 된다. 문제는 인코딩의 일치 문제는 서버와 클라이언트의 양쪽에 걸쳐서 생기는 문제이므로, 브라우저에게 특정 인코딩을 강제할 수 없다는 점이다. 이런 문제는 웹 문서 내에서 다국어 식별자가 링크로 사용되는 경우에 호환성에 문제가 발생할 수 있다는 점에서 데이터 통합 등을 화두로 하고 있는 웹 2.0에서는 문젯거리가 된다. 특히 이러한 경우에는 외부의 특정 코드를 웹 서버에 추가하지 않고서는 해결할 수 있는 방법이 없다.



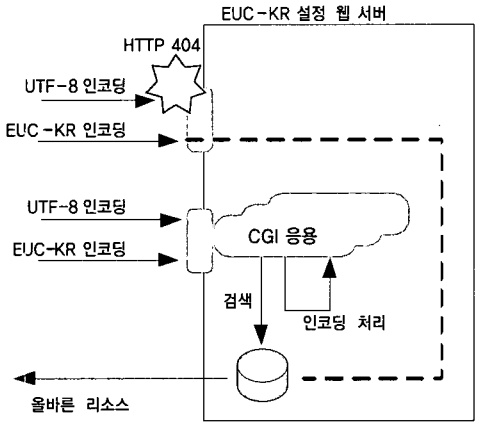
(그림 1) 브라우저와 웹 서버의 인코딩 정합

4.2. Cool URI의 예를 통해 본 인코딩 혼재의 해결책

다국어 식별자의 유용성과 인코딩 혼재에 따른 문제점을 보이기 위해 Cool URI에 대해서 살펴보도록 하자. Web 2.0의 주요 기술 중의 하나인 Cool URI[14]는 웹 리소스에 영속적인 인터넷 식별자를 제공하는 기술이다. Cool URI는 블로그(Blog)의 퍼머링크(PermaLink)나 트랙백(Trackback)에서 중요하게 사용되는 기술이며, 데이터 결합 시의 링크 불일치를 해결해준다. 일반적인 인터넷 식별자는 시스템에 종속적인 구조로 사용자가 기억하기 어렵거나 시스템의 설정 변화에 의해서 변경된다. 예를 들어 "http://www.foobar.com/blog?id=82728"와 같은 인터넷 식별자는 사용자가 쉽게 기억하기 어려우며, 또한 식별자의 일부분이 데이터베이스에서 임의로 정해진 구조이기 때문에 데이터베이스를 갱신하는 경우, 대부분의 인터넷 식별자가 변경되게 된다. 따라서 이런 임시적인 인터넷 식별자를 퍼머링크를 이용한 데이터 결합 시에 사용하게 되면, 데이터 일관성을 보장할 수 없게 된다. 이에 비해 Cool URI는 자연어에 맞는 구조를 갖고 있으며, 시스템의 변동에 따라서 인터넷 식별자가 변화되지 않는다. "http://www.foobar.com/my_first_article"과 같은 식별자는 이해하기도 편하며, 시스템에 종속되지 않는 인터넷 식별자이므로 데이터 결합 등에 유용하게 사용될 수 있다. 그러나 Cool URI도 인터넷 식별자이기 때문에 현재는 US-ASCII 만을 이용하도록 되어 있다. 따라서 국내에서 Cool URI를 사용하기 위해서는 불편하더라도 영문으로 변환해서 사용해야 한다. 그러나 이는 모호성을 가중시키며 불편함을 가중시킨다. 예를 들어 "한글"이란 Cool URI를 사용하고 싶으면, Hangul로 할지, Hangoul로 할지는 개인의 생각에 따라서 바뀔 수 있고, 이러한 Cool URI는 같은 리소스에 대해서 다른 식별자를 제공하기 때문에 혼란을 야기하게 된다.

현재 국내에서는 Cool URI의 필요성을 인지하고, 태터툴즈와 MT 플러그인과 같은 블로그용 Cool URI의 지원이 이루어지고 있다. 이는 매우 바람직한 현상이나 내부적으로는 이 방식도 문제점을 갖고 있다. 기존 애플리케이션들에서 제공하는 대부분의 Cool URI는 실제로는 EUC-KR이나 UTF-8으로

인코딩된 URI 스트림을 애플리케이션 내부적으로 인코딩 방식을 자동으로 파악하여 프로그래밍적으로 처리하는 것이지 표준안에 근거한 다국어 식별자의 기능은 아니다. 즉, (그림 2)에서 볼 수 있는 것처럼 인코딩이 혼재된 상황에서 Cool URI를 지원하기 위해서는 반드시 프로그램 내부에서 인코딩을 추측하고, 이를 처리하기 위한 코드가 준비되어야 한다.



(그림 2) 인코딩 혼재시의 처리 방안

이러한 처리 방안은 개별 애플리케이션에서는 가능할지 모르지만 근본적인 해결책은 될 수 없으며, (그림 2)에서 볼 수 있는 것처럼 Cool URI가 웹 서버만을 이용한 페이지 결합 시에는 문제를 발생시킬 수 있게 된다는 것을 의미한다. 따라서 Cool URI가 특정 블로그 내에서 운용은 가능하지만, 이 Cool URI를 다른 페이지의 일부로 사용하거나 하는 경우에는 호환성의 문제가 발생할 수 있다. 이런 문제를 근본적으로 해결하기 위해서는 웹 페이지와 서버의 기본 인코딩 설정을 UTF-8을 사용하도록 권고하고, 이를 통하여 다국어 식별자가 제대로 해석될 수 있도록 해야 할 것이다.

V. 결 론

웹 2.0과 시맨틱 웹이 대두되면서 웹의 새로운 도약에 대한 기대가 널리 퍼지고 있다. 웹 2.0은 산업계에서 새로운 비즈니스의 동력으로 인정받고 있으며, 시맨틱 웹은 웹 상에서의 자동화된 데이터 추론과 지능형 동작을 가능하게 할 것으로 예측된다. 이러한 차세대 웹 환경에서 리소스에 대한 인터넷 식별자의 중요성은 더욱 커지게 되는데, 웹 2.0은 UCC(User Creative Contents)나 블로그의 결합을 위해서 인터넷 식별자가 중요하며, 시맨틱 웹에서는 리소스와 술어를 지정하는데 기본이 된다. 이러한 상황에서 모국어/자연어를 인터넷 식별자에 사용하는 것은 차세대 웹의 활성화에 매우 중요한 기술적 요건이라 할 수 있다.

본 기고에서는 모국어/자연어를 식별자에 사용할 수 있는 다국어 식별자의 표준화 현황과 국내의 실제 적용 방안에 관해서 살펴보았다. 다국어 식별자가 네트워크에서 사용되기 위해서는 UTF-8 인코딩을 통한 인터넷 식별자로의 변환이 중요하다. 그러나 현재 국내의 웹 환경은 EUC-KR과 UTF-8 인코딩이 혼재되어 있는 상황이기 때문에 다국어 식별자의 적용에 현실적인 어려움이 있다. 다행히도 다국어 식별자를 쉽게 적용할 수 있는 다양한 웹 2.0 애플리케이션들이 나오고 있지만, 애플리케이션 내부에서만 다국어 식별자가 지원된다는 한계점을 갖고 있다. 다국어 식별자의 도입은 파일 이름이 한글로 되어 있다고 웹 브라우저에서 열 수 없다는 단순한 문제를 넘어서, 데이터 결합과 리소스의 지정(identification)이 중요한 차세대 웹에서 반드시 지원되어야 하는 기초적인 기술이다. 따라서 국내의 웹 커뮤니티에서 UTF-8 인코딩과 다국어 식별자의 활용을 적극적으로 한다면 국내 웹 기술과 정보통신 산업 발전에 큰 기여를 할 것으로 기대된다.

[참 고 문 헌]

[1] Tim, B-L., James, H., and Lassila, O, "The Semantic Web", Scientific American, Vol 284, No.5, pp.34-43, 2001

[2] Tim, O., "What is Web 2.0; Design Patterns and Business Models for the Next Generation of Software", O'Reilly Network, http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, Aug. 2006

[3] Dion, H, "The State of Web 2.0", Web Service Journal, http://web2.wsj2.com/the_state_of_web_20.htm, Aug. 2006-11-06

[4] Tim, B-L. and et. al., "Uniform Resource Identifier (URI): Generic Syntax", IETF RFC 3986, 2005

[5] Bo, L., "The Semantic Web", Wiley, 2005

[6] Tim, B-L., Robert, C., Ari, L., Henrik, F.N., Arthur, S., "The World-Wide Web", CACM, Vol.37, No.8, pp.76-82, 1994

[7] The Unicode Consortium, "The Unicode Standard, Version 4.0", 2003.

[8] Duerst, M., "Internationalized Resource Identifier", IETF RFC 3987, 2005.

[9] Yergeau, F. "UTF-8, a transformation format of ISO 10646", IETF RFC 3629, 2003.

[10] Fielding, R., "Hypertext Markup Language -- HTTP/1.1", IETF RFC 2616, 1999.

[11] Davis, M. and M. Duerst, "Unicode Normalization Forms", Unicode Standard Annex #15, Apr. 2003.

[12] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[13] Costello, A., "Punycode: A Bootstring encoding of Unicode for use with Internationalized Domain Names in Applications (IDNA)", RFC 3492, March 2003.

[14] Tim, B-L.,"Cool URIs don't change", W3C style guide, 1998.

정의현

1992년 한양대학교 전자공학과 학사
1994년 한양대학교 전자공학과 석사
1999년 한양대학교 전자공학과 박사
1999년 ~ 2002년 대우통신 선임연구원
2002년 ~ 2003년 SCT 연구소장
2004년 ~ 현재 안양대학교 디지털미디어학부 교수
관심분야 : 시맨틱 웹, 센서 네트워크, 웹 2.0