韓國 컴퓨터情報學會 論文誌
第11卷 第5號, 2006. 11.

2006-11-5-2-8

웹 2.0을 위한 다국어 식별자 기반의 Cool URI에 대한 연구

정의현*, 김 원**, 송관호***, 박찬기****

A Research on Cool URI based on Internationalized Resource Identifier for Web 2.0

Eui-Hyun Jung*, Weon Kim**, Kwan-Ho Song***, Chan-Ki Park****

요 약

차세대 웹은 표현 중심의 웹을 데이터 중심의 웹으로 이끌 것으로 예측되며, Web 2.0과 시맨틱 웹의 기술적 결합이 될 것이다. 차세대 웹은 시맨틱 처리, 웹 플랫폼과 데이터 결합이 매우 중요한 기술적 요소이다. 이 중에서 데이터 결합에 사용되는 Cool URI는 영속적이고 사용자 친화적인 URI를 제공하는 기술이며, 이미 블로그 등에서 매우 중요하게 사용되고 있다. 그러나 Cool URI는 한글과 같은 다국어 환경에 적합하도록 구성되어 있지 않으며, 여러 인코딩이 혼재된 국내 웹 환경에서는 쉽게 사용하기 어려운 상황이다. 본 논문에서는 이러한 Cool URI를 다국어 식별자와 같이 사용하기 위한 기술적인 고려 사항 및 Cool URI 웹 컴포넌트에 관하여 논한다. 제시한 방식은 인코딩의 종류에 상관없이 동일한 기능을 제공하며, 다른 애플리케이션에서 쉽게 사용 가능하도록 파일 시스템 기반과 CGI 기반 방식을 모두 지원한다. 여러 환경에서 실험한 결과는 구현된 웹 컴포넌트가 설계 목표를 만족함을 보여주었다.

Abstract

Web 2.0 and Semantic Web technology will be merged to be a next generation Web that leads presentation-oriented Web to data-centric Web. In the next generation Web, semantic processing, Web platform, and data fusion are most important technology factors. Among them, Cool URI used for data fusion can provide permanent and human readable URI and it has been already used in Blog system. However, Cool URI is not suitable to I18N environment such as Hangul and it is difficult to be adopted because several encodings are mingled in Korea Web society. In this paper, we discussed technology issues and Cool URI Web component to use Cool URI with Internationalized Resource Identifier (IRI). A proposed approach provides same function regardless of encoding type and supports both file system based and CGI based methods to be easily used from other applications. Results from several test environments show the implemented Web component satisfy the design purpose.

▸ Keyword : Web 2.0, URI, IRI, Unicode

• 제1저자 : 정의현
• 접수일 : 2006.10.19, 심사일 : 2006.10.27, 심사완료일 : 2006.11.18
* 안양대학교 디지털미디어공학 전임강사 ** 인터넷 진흥원 단장 *** 인터넷 진흥원 원장 **** 인터넷 진흥원 팀장
※본 연구는 2006년도 한국 인터넷 진흥원 공동 연구과제의 결과물임

Ⅰ. 서 론

웹(Web) 기술은 정보시스템 전반에 널리 사용되는 기술로서 사용자에게 정보를 효과적으로 전달하는데 가장 유효한 기술로 평가된다[1]. 이는 HTML(Hyper Text Markup Language)과 같은 표현(presentation) 중심의 마크업(mark-up) 언어와 실제 HTML을 렌더링(rendering)하는 브라우저(browser)가 분리되어 있는 구조를 갖고 있기 때문이다. 이러한 구조에서는 렌더링 클라이언트의 종류에 상관없이 내용을 어떻게 표현할 것인지에 대해서 결정한 후 HTML을 이용하여 문서를 작성하면 된다. 그렇기 때문에 현재의 웹 기술을 이용하면, 컨텐츠 저작자들이 복잡한 렌더링 기술에 대한 고려 없이 손쉽게 HTML 기반의 컨텐츠를 작성하는 것이 가능하며, 표현력이 풍부하면서 간단한 HTML 문서양식 때문에 많은 오프라인 컨텐츠가 온라인에 이전되어, 현재의 인터넷은 웹을 기반으로 하는 풍부한 정보 자원을 갖게 되었다[2].

이렇듯 기존의 웹이 인간(human) 사용자에게 정보를 전달하는데 성공적인 기술이었던데 반해, 기계간의 연동이나 데이터 결합 (data fusion) 등의 문제에서는 약점을 노출하고 있다[3]. 이것은 크게 두 가지 문제로 요약할 수 있는데, 첫째는 HTML 자체가 표현 중심의 언어라는 점이다. HTML 자체는 문서의 내용(semantic)을 표현할 수 있는 방법을 갖고 있지 않다. 이런 문제 때문에 기계에게 의미를 전달할 수 있는 새로운 구조인 시맨틱 웹(Semantic Web)[4]이 새롭게 주목을 받고 있다. 두 번째 이유는 HTML의 물리적인 위치를 나타내는 URL(Uniform Resource Locator)[3]이 영속적인(permanent) 값을 나타내는 것이 아니고, 파일 시스템이나 데이터베이스에서 만들어진 상대적이고 임시적인 값을 갖고 있다는 점이다. 이러한 URL의 위치 변동은 데이터 결합 시에 링크 변동에 의한 데이터 일관성(consistency)의 문제를 야기하게 된다[5][6].

최근에 대두된 웹 2.0[7] 은 이러한 표현 중심의 웹에서 데이터 결합이나 기계간의 연동, 웹 플랫폼에 대한 요구 사항을 충족시키기 위한 기술적 흐름이다. 웹 2.0의 주요한 기술 중의 하나인 Cool URI[6]는 웹 리소스에 영속적인 URI(Uniform Resource Identifier)를 제공하는 기술이다. Cool URI는 블로그(Blog)의 퍼머링크(PermaLink)나 트랙백(Trackback)에서 중요하게 사용되는 기술이며, 데이터 결합 시의 링크 불일치를 해결해준다.

일반적인 URI는 시스템에 종속적인 구조로 사용자가 기억하기 어렵거나 시스템의 설정 변화에 의해서 변경된다.

예를 들어 "http://www.foobar.com/blog?id=82728"와 같은 URI는 사용자가 쉽게 기억하기 어려우며, 또한 URI의 일부분이 데이터베이스에서 임의로 정해진 구조이기 때문에 데이터베이스를 갱신하는 경우, 대부분의 URI가 변경되게 된다. 따라서 데이터 결합 시에 이런 임시적인 URI를 사용하게 되면, 데이터 일관성을 보장할 수 없게 된다. 이에 비해 Cool URI는 자연어에 맞는 구조를 갖고 있으며, 시스템의 변동에 따라서 URI가 변화되지 않는다. "http://www.foobar.com/my_first_article"과 같은 URI는 이해하기도 편하며, 시스템에 종속되지 않는 URI이므로 데이터 결합 등에 유용하게 사용될 수 있다.

그러나 Cool URI를 US-ASCII 외의 문자집합에서 이용하는 것은 문제점을 내포하고 있다. 이는 URI 자체가 현재 US-ASCII 만을 유효한 문자집합으로 사용하도록 규정되어 있기 때문이다. 따라서 한글로 Cool URI 기능을 사용하기 위해서는 다국어 식별자 (IRI: Internationalized Resource Identifier)[8] 표준안에 따라서 구성되도록 시스템 내부에서 구축되어야 한다. 다국어 식별자는 IETF 표준안으로 유니코드(Unicode)[9]를 URI 문자열에 넣을 수 있도록 구성된 표준이다. 그러나 현재 국내의 웹은 다국어 식별자 표준안이 준수되지 않고, EUC-KR[10]과 UTF-8[11] 인코딩 방식이 혼재되어 사용되는 상황이다. 비록 태터툴즈와 같은 블로그 툴이 한글화된 Cool URI를 자신의 코드 내에서 제공하고 있지만, 시스템 내부에서만 독자적인(proprietary) 방식으로만 제공되고 있다는 한계점을 갖고 있다. 차세대 웹에서 Cool URI가 여러 애플리케이션에서 원활히 사용되기 위해서는 재사용이 가능한 웹 컴포넌트 형태로 Cool URI가 제공되어야 한다.

본 논문에서는 인코딩이 혼재된 상황에서도 동일하게 동작하는 다국어 식별자가 고려된 Cool URI의 기술적 요구와 그 해결 방안에 관하여 논한다. Cool URI는 파일 시스템 기반과 CGI 기반으로 지원할 수 있는데, 본 논문에서는 두 개의 방식을 모두 지원할 수 있도록 구성된 웹 컴포넌트의 설계 방안과 구현에 관하여 설명한다. 구현된 웹 컴포넌트는 PHP 모듈로 구현되어 있어, 기존의 PHP 기반의 웹 시스템에서 Cool URI를 적용하고자 하는 경우 유연하게 사용할 수 있을 것으로 기대된다.

본 논문은 다음과 같은 구성을 갖는다. 먼저 2장에서는 다국어 식별자 기반의 Cool URI를 지원하기 위한 기술적 요구 사항들에 대해서 분석한다. 3장에서는 제안하는 웹 컴

포넌트의 구조에 대해서 설명하고, 4장에서는 설계된 웹 컴포넌트를 실제 사용하여 다국어 식별자 기반의 Cool URI의 효용에 대해서 살피고, 5장에서 결론을 맺는다.

II. 기술적 고려사항

2.1 다국어 식별자의 문자 인코딩

URI 표준인 RFC 3986[12]에 의하면 URI는 US-ASCII 문자 집합만을 이용하도록 규정되어 있다. 따라서 한글을 URI에 넣는 것은 원칙적으로 표준에 어긋난다. 그러나 URI로 인터넷 자원(resource)을 나타내는데 모국어를 이용하는 것이 이해하기 편하고, 적용하기 편하며 기억하기 편리하다. 예를 들어, "http://www.example.org/친구_얼굴" 이라는 URI가 친구 얼굴에 대한 인터넷 자원을 나타내는 것이라면, 이 URI가 무엇을 의미하는지는 글을 읽을 수 있는 대한민국 사람이라면 누구나 알 수 있다. 그러나 이것을 기본적인 URI 표준에 맞추기 위해서는 "http://www.example.org/friend_face"나 아예 의미 없는 "http://www.example.org/aaa_111"로 변환하여 사용해야 한다. 이것은 URI가 가리키는 자원에 대한 모호성(ambiguity)을 가중시키며, URI의 효용성을 감소시킨다. 이러한 문제점을 해소하기 위하여 다양한 문자 집합을 URI에 사용할 수 있는 다국어 식별자 표준안이 RFC 3987[8]로 제안되었다. 다국어 식별자는 표 1과 같이 다양한 문자 집합을 URI에 사용할 수 있는 특징을 갖고 있다.

표 1. 다국어 식별자의 예
Table 1. Examples of IRI

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

다국어 식별자는 유니코드를 사용하도록 규정되어 있기 때문에 브라우저에서 웹 서버로 전송되기 위해서는 적절한 문자 인코딩을 거쳐야 한다. 이는 유니코드는 논리적인 문자 체계이기 때문에 물리적으로 저장되거나 전송 시에는 문자 인코딩을 하도록 되어 있기 때문이다. 다국어 식별자의 표준안에서는 유니코드로 표현된 다국어 식별자는 전송 시에 반드시 UTF-8 인코딩으로 전송하도록 규정되어 있다[8]. 그러나 국내의 웹에서 문자 인코딩으로 사용되는 문자 인코딩 방식은 UTF-8과 EUC-KR 표준으로 나누어져 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 2. 다국어 식별자의 인코딩 표현
Table 2. Encoding representation of IRI

방식	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

표 2에서 볼 수 있는 것처럼, 동일한 다국어 식별자가 문자 인코딩 방식에 따라서 다른 형태의 바이트 스트림으로 서버에 전송되게 된다. 문제는 웹 서버 측에서 사전에 지정된 동일한 문자 인코딩으로 전송되지 않은 URI 스트림에 대해서는 실제 리소스의 위치를 파악할 수 없다는 점이다.

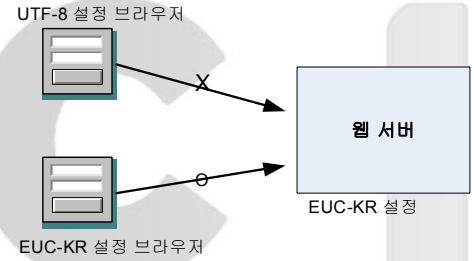

그림 1. 브라우저와 웹 서버의 인코딩 정합
Fig 1. Encoding matching between server and browsers

그림 1에서 볼 수 있는 것처럼 EUC-KR을 기본 문자 집합으로 지정한 웹 서버에게 웹 브라우저의 인코딩 방식에서 UTF-8 옵션으로 지정하여 URI를 전송하게 되면, URI에 해당하는 리소스에 접근할 수 없기 때문에 웹 서버는 HTTP 404 에러를 발생시키게 된다. 그러나 UTF-8 옵션을 해제한 후에 전송하게 되면, 제대로 페이지가 렌더링 되게 된다. 문제는 인코딩의 일치 문제는 서버와 클라이언트의 양쪽에 걸쳐서 생기는 문제이므로, 브라우저에게 특정 인코딩을 강제할 수 없다는 점이다.

2.2 Cool URI의 적용 구조

앞 절에서 살펴본 바와 같이 다국어 식별자를 사용하지 않고, US-ASCII 만을 이용하여 Cool URI를 사용한다면

이러한 인코딩 문제는 문제될 것이 없다. 그러나 Cool URI의 기본적인 목표가 영속적(permanent)이고 사용자 친화적인(human readable) URI를 구성하는 것이 주요한 목표이기 때문에 모국어가 지원되는 다국어 식별자의 활용은 필수적이다. 다국어 식별자의 활용을 위한 기술적 요구 사항은 크게 2가지이다.

첫째, 인코딩 방식이 혼재된 상황에서도 동일한 동작을 보장해주어야 한다. 현재 국내 웹 환경에서는 문자 인코딩 방식이 혼재되어 사용되기 때문에 외부 브라우저가 UTF-8을 사용하든, EUC-KR을 사용하든 다국어 식별자 표준에 근거한 동일한 Cool URI 기능을 투명하게(transparent) 이용할 수 있도록 해야 한다. 이를 위해서는 웹 서버의 기본 인코딩 방식으로 해석되지 않는 인코딩에 대해서 서버가 대응할 수 있는 방법을 제공해야 한다.

둘째, Cool URI의 적용 방안에 동일한 서비스를 제공하는 웹 컴포넌트 구조이어야 한다. 기본적으로 Cool URI는 파일 시스템 기반 방식과 CGI 기반 방식으로 나눌 수 있다. 이 두 가지 방식은 인코딩의 처리 주체가 다르다는 특징을 갖고 있다. 파일 시스템 기반 방식은 웹 서버가 URI 스트림에 대한 문자 인코딩에 대한 처리 권한을 갖기 때문에 시스템에 좀 더 종속적인 형태를 갖게 된다. 이에 비해 CGI 기반 방식은 애플리케이션마다 해당 스트림에 대한 인코딩을 갖기 때문에 통일된 방식으로 인코딩을 처리하기 어렵다는 문제점을 갖고 있다. 애플리케이션들이 다국어 식별자 기반 Cool URI를 원활히 적용하기 위해서는 애플리케이션들이 어떤 방식을 채택하더라도 사용할 수 있는 웹 컴포넌트 구조가 되어야 한다.

파일 시스템 기반 Cool URI는 URI 생성 시에 파일 시스템에서 폴더나 파일명이 URI로 주어지는 경우이다. 예를 들어, "http://www.foobar.com/한글소개"라는 URI가 주어지면, 내부에서 "한글소개"라는 파일이 만들어지게 된다. 파일 시스템으로 Cool URI가 구성된 경우는 웹 서버가 URI에 대응되는 처리를 맡게 된다. 따라서 웹 서버가 일차적인 URI 인코딩에 대한 처리를 하게 된다. 이러한 경우에 웹 서버가 URI 인코딩을 웹 서버의 기본 인코딩 방식으로 처리하기 때문에 브라우저에서 전송된 인코딩 방식과 정합되지 않는 경우, HTTP 404 에러[13]를 발생시키게 된다. 파일 시스템 기반 Cool URI는 인코딩의 주체가 웹 서버이기 때문에 애플리케이션 개발자는 손 쓸 방법이 없다.

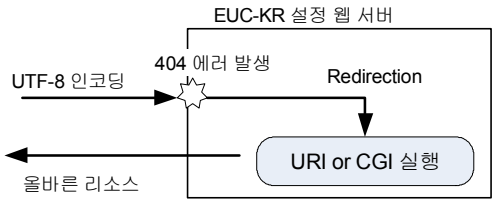

그림 2. 404 에러의 처리
Fig 2. Processing of 404 error

이를 해결하기 위해서는 그림 2에서처럼 404 에러가 발생하는 경우에 대해서, 리다이렉션(redirection) 기법을 통해서 해결해주어야 한다. 리다이렉션 기법이란 404 에러가 발생하는 경우에 자동으로 다른 URI로 위치를 옮기거나, 정해진 CGI 코드를 실행하는 방법이다.

아파치 웹 서버에서 리다이렉션 기법을 사용하는 데는 ErrorDocument를 이용하는 방식과 mod_rewrite 모듈을 이용하는 방식이 있다. mod_rewrite는 URI 리다이렉션에 필요한 대부분의 기능을 제공하며, 재작성 패턴 (rewrite pattern)을 받아들여 URI에 적용을 한다. 여기서 패턴이라는 것은 정규 표현식 (regular expression)을 의미하며, 이 정규 표현식에 일치가 되는 패턴을 요청한 URI에 출력하는 것이다. 이 기능은 상당히 유용하여, 가상 호스트의 작성이나 URL 재작성 등에 매우 유용하게 사용된다. 예를 들어, http://domain.com/~user_id 의 경우에 mod_rewrite를 이용하면, http://user_id.com 으로 바꿀 수 있다. mod_rewrite를 이용하면 거의 대부분의 URI 리다이렉션 기능을 수행할 수 있기 때문에 유연한 Cool URI의 지원이 가능한 특징을 갖고 있다. 그러나 mod_rewrite는 확장 모듈을 설치해야 하고 복잡한 정규 표현식을 구성해야 하는 단점이 있다. 이에 비해 ErrorDocument는 아파치 웹 서버의 기본 기능이므로 아파치 웹 서버가 설치된 곳에서는 어디에서나 사용 가능하며, 사용법도 매우 직관적이다. 따라서 ErrorDocument를 사용하는 것이 파일 시스템 기반 Cool URI에서는 적절한 방식이 될 수 있다.

CGI 기반 방식은 Cool URI의 처리를 CGI가 담당하는 것이다. 이 경우는 웹 서버가 다국어 식별자의 인코딩을 담당하는 것이 아니기 때문에, 매우 융통성 있게 처리할 수 있다.

예로 "http://www.com/index.cgi?id=한글소개"라는 Cool URI가 있다면, 해당 Cool URI의 "한글소개"라는 부분은 index.cgi가 인코딩을 판단하여 처리하게 된다. 물론 이런 형태의 구조는 Cool URI의 형태에 적합하지 않지만, mod_rewrite 기능을 결합하면, 외부에는 폴더나 파일명으

로 보여지고, 내부적으로는 CGI 처리를 할 수 있으므로, 파일 시스템 기반 Cool URI와 외부에서는 구분할 수 없게 된다. 이러한 처리의 유연함 때문에 기존 블로그(blog)나 게시판 시스템 등이 많이 채택하는 방법이지만, 독자적으로 코드를 만들어 사용하기 때문에 코드 재사용성은 거의 없다. 따라서 Web 2.0에 대응되는 애플리케이션들이 쉽게 Cool URI를 사용하기 위해서는 적절한 웹 컴포넌트 형태의 구조가 제공되어야 한다.

다국어 식별자 기반의 Cool URI에 대한 기술적 요구사항을 한마디로 정리하면, 인코딩 종류와 기술 방식에 상관없이 동일한 Cool URI 기능을 제공해야 하며, Cool URI를 적용하고자 하는 애플리케이션들이 쉽게 사용할 수 있도록 웹 컴포넌트 형태를 가져야 한다는 점이다.

III. 웹 컴포넌트 구조

설계된 웹 컴포넌트는 파일 시스템 기반과 CGI 기반의 Cool URI 모두에 대해서 동일하게 작동하도록 설계되었다. 이를 위해서 웹 컴포넌트는 파일 시스템 기반 방식에서는 독립적인 모듈로 동작할 수 있어야 하고, CGI 기반 방식에서는 다른 모듈에게 기능을 제공하는 부속 모듈 형태가 되어야 한다. 본 논문에서는 국내에서 가장 많이 사용되는 웹 플랫폼인 PHP와 아파치(Apache) 웹 서버를 기준으로 웹 컴포넌트를 설계하였다.

3.1 파일 시스템 기반의 Cool URI

논문에서는 파일 시스템 기반의 Cool URI를 지원하기 위해서 아파치의 기본 기능인 ErrorDocument를 이용하였다. ErrorDocument를 사용하는 경우 아파치 서버의 httpd.conf에 지시자를 넣는 경우도 있지만, 아파치 서버의 설정 파일은 관리자 권한을 갖고 있어야 하므로, Cool URI를 적용하고자 하는 폴더에 .htaccess 파일을 다음과 같이 작성해준다. 여기서는 itest라는 폴더가 Cool URI를 지원하는 폴더라고 가정하였다.

```
ErrorDocument 404 /itest/iri_support.php
ErrorDocument 403 /itest/iri_support.php
```

이렇게 ErrorDocument 지시자가 지정된 경우에는 itest 폴더 밑에 Cool URI에 대응되는 파일명에 대해서 인코딩이 안 맞아서 리소스를 못 찾는 경우에는 iri_support.php 파일이 수행되게 된다. iri_support.php는 들어오는 데이터 스트림의 바이트 패턴을 파악하여, EUC-KR과 UTF-8의 인코딩을 판단하여 UTF-8인 경우에는 자동으로 ECU-KR 인코딩으로 바꾸어 웹 서버에게 다시 전달하게 된다.

iri_support.php는 크게 2개의 함수와 1개의 실행 코드로 구성되어 있다. 첫 번째 함수는 텍스트의 바이트 패턴을 통해서 인코딩을 파악하는 함수인 encoding()이다. UTF-8의 경우에는 절대 나올 수 없는 바이트 패턴이 있고, 이 패턴을 이용해서 인코딩을 파악하는 함수이다.

```
function encoding($text) {
  if(preg_match('/^([\x09-\x7E])*$/x', $text)) {
    return 'us-ascii';
  } else if(preg_match('/^([\x09-\x7E]|[\xC2-\xDF][\x80-\xBF]|
    \xE0[\xA0-\xBF][\x80-\xBF]|[\xE1-\xEC\xEE\xEF][\x80-\xBF]{2} |
    \xED[\x80-\x9F][\x80-\xBF]|\xF0[\x90-\xBF][\x80-\xBF]{2} |
    [\xF1-\xF3][\x80-\xBF]{3}|\xF4[\x80-\x8F][\x80-\xBF]{2})*$/x', $text)) {
      return 'utf-8';
  } else if(preg_match('/^([\xA1-\xFE][\x09-\x7E])*$/x', $text)) {
      return 'euc-kr';
  }
}
```

두 번째 함수는 UTF-8 인코딩 문자를 받아서 EUC-KR 인코딩 문자열로 바꾸어주는 함수인 utf8_to _euckr()이다.

```
function utf8_to_euckr($text) {
  $src_encoding = "utf-8";
  $target_encoding = "euc-kr";
  $target_text = iconv($src_encoding,
                       $target_encoding, $text);
  return $target_text;
}
```

최종적으로 실행되는 코드는 다음과 같다. 이 코드에서는 404에러에 의해서 리다이렉션된 URI의 인코딩을 파악하여, UTF-8인 경우에는 EUC-KR로 자동으로 변경하여 리소스를 찾고, 리소스가 존재하는 경우 브라우저로 전송하게 된다. 여기서는 서버의 기본 인코딩 설정을 EUC-KR로 가정하였다.

```
$prev_uri = $_SERVER["REDIRECT_URL"];

$code_type = encoding($prev_uri);
if($code_type == 'utf-8') {
  $converted_uri = utf8_to_euckr($prev_uri);
  if($converted_uri != false) {
    header("Location: ".$converted_uri);
    for($i=0;$i < 512; $i++) {
      echo(' ');
    }
  }
} else {
  echo("404: 해당 파일을 찾을 수 없습니다.");
}
```

이 코드의 흐름은 그림 3에서 볼 수 있는 것처럼 서버가 EUC-KR로 인코딩이 지정된 경우, 브라우저의 인코딩이 UTF-8을 사용하여 서버에서 404 에러가 나온 경우에 iri_support.php 파일이 자동적으로 수행되게 된다. 그리고 이렇게 iri_support.php가 404 에러에 의해 수행되면, iri_support.php에 의해서 자동으로 해당 파일을 찾게 된다.

참고로 마이크로소프트의 IIS(Internet Information Server)를 사용하는 경우에는 IIS 메타베이스 속성 중에서 HttpErrors 속성에 iri_support.php 파일을 연결하면 같은 결과를 얻을 수 있다.

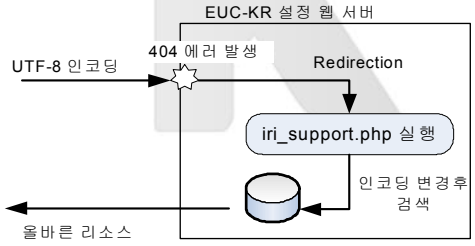

그림 3. iri_support.php의 인코딩 처리
Fig 3. Encoding processing of iri_support.php

3.2 CGI 기반의 Cool URI

CGI 기반의 Cool URI에서는 URI 스트림이 웹 서버에 도착하면, 웹 서버는 해당 애플리케이션의 CGI 모듈에 인자들에 대한 처리를 넘겨주게 된다. 따라서 애플리케이션들이 인코딩의 주체가 되므로 웹 컴포넌트는 애플리케이션들이 호출 가능한 코드를 제공해주어야 한다. 이런 구조로 가게 되면, 그림 4에서 볼 수 있는 것처럼, 애플리케이션들은 해당 인코딩에 맞추어 제공된 iri_support.php의 적절한 함수를 호출하여 인코딩 처리를 할 수 있게 된다.

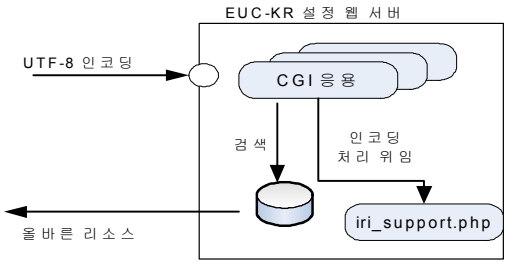

그림 4. CGI 응용의 인코딩 처리 위임
Fig 4. Delegation of encoding processing from CGI applications

본 논문에서는 앞에서 작성한 문자 인코딩을 추측할 수 있는 encoding() 함수와 문자열 변환 방식인 utf8_to_euckr()외에 EUC-KR 인코딩에서 UTF-8으로 인코딩을 바꿔주는 euckr_to_utf8() 함수를 iri_support.php 모듈에 추가하였다. 이 함수의 형태는 다음과 같다

```
function euckr_to_utf8($text) {
  $src_encoding = "euc-kr";
  $target_encoding = "utf-8";
  $target_text = iconv($src_encoding,
                       $target_encoding, $text);
  return $target_text;
}
```

이 함수들을 이용하여 URI 스트림을 바꾸는 샘플 코드는 다음과 같다. 여기서는 테스트를 위해 인자로 val이라는 인자 명에 대해서 호출한 것으로 가정하고, 서버의 기본 인코딩은 UTF-8을 가정하였다. 제시된 샘플 코드는 iri_support.php를 사용하고자 하는 애플리케이션에서 서버의 기본 인코딩에 맞추어 적절히 수정하여 사용할 수 있다.

```
$val = $_GET["val"];
if($code_type == 'euc-kr') {
  $converted_arg = euckr_to_utf8($val);
} else {
  $converted_arg = $val;
}
return $converted_arg;
```

샘플 코드에서는 단순히 Cool URI에 대응되는 인자의 인코딩만을 서버 설정대로 바꾸었지만, 앞에서 설명한 것처럼 mod_rewrite를 사용하게 되면, CGI 코드가 외부에서는 숨겨지게 되어 Cool URI로서의 기능을 가질 수 있게 된다.

Ⅳ. 실험 및 효용성 평가

본 논문에서 제시한 방식이 다국어 식별자 기반의 Cool URI의 기술적 요구사항을 만족할 수 있는지 확인하기 위하여 다음과 같이 실험 준비를 하였다. 먼저 구현된 Cool URI 웹 컴포넌트가 제대로 동작하는지 확인하기 위하여 다음과 같이 실험을 하였다. 먼저 /itest 폴더와 /xtest 폴더를 만들고, 각 폴더에 "한글"이라는 파일을 만들었다. 여러 인코딩에 대해서 동일한 Cool URI를 제공하는 것은 "http://www.raccooncity.co.kr/itest/한글"이고, 제대로 제공하지 못하는 URI는 "http://ww w.raccooncity.co.kr /xtest/한글"로 지정하였다. 이 두 개의 URI에 대해서, 웹 브라우저에서 UTF-8과 EUC-KR을 기반 인코딩으로 하여 각 4가지 경우를 실험하여 파일 시스템 기반 Cool URI를 테스트하였다. CGI 기반 방식의 구현을 테스트하기 위해, 다국어 식별자를 지원하는 ipass.php와 지원하지 않는 xpass.php를 테스트하였다. 서버의 기본 설정은 아파치 기본 설정인 EUC-KR을 이용하였다. 테스트 환경과 그 결과는 표 3과 같다.

표 3. 테스트 환경 및 결과
Table 3. Test environment and results

테스트 URI	브라우저 인코딩	성공여부
/itest/한글	EUC-KR	O
/itest/한글	UTF-8	O
/xtest/한글	EUC-KR	O
/xtest/한글	UTF-8	X
/ipass.php?val=한글	EUC-KR	O
/ipass.php?val=한글	UTF-8	O
/xpass.php?val=한글	EUC-KR	O
/xpass.php?val=한글	UTF-8	X

파일 시스템 기반 방식은 그림 5에서 볼 수 있는 것처럼, 서버가 EUC-KR로 설정되어 있는 경우에는 UTF-8 인코딩을 지정하고, 다국어 식별자 Cool URI 처리가 되어 있지 않은 /xtest 폴더의 경우에는 404 에러가 발생함을 알 수 있다. 이에 비해 다국어 식별자 Cool URI 처리가 되어 있는 /itest의 경우에는 자연스럽게 해당 리소스를 찾아서 다국어 식별자 기반의 Cool URI로 동작함을 알 수 있다.

그림 5. 테스트 결과 화면
Fig 5. Snapshots of test results

Ⅴ. 결론 및 향후 과제

Cool URI는 웹 2.0과 같은 차세대 웹에서 데이터 결합의 용도로 중요하게 사용될 것으로 기대된다. Cool URI에 한글을 사용하기 위해서는 다국어 식별자를 이용해야만 하지만, 인코딩의 혼재 때문에 통일된 다국어 식별자 사용이 어려운 환경이다. 이러한 문제 때문에 애플리케이션별로 독자적인(proprietary) 방식으로만 한글 Cool URI를 지원할 뿐, 여러 애플리케이션에서 사용 가능한 웹 컴포넌트에 대해서는 연구가 되지 않았다. 본 논문에서는 다국어 식별자 기반의 Cool URI 지원을 위해서 파일 시스템 기반과 CGI 기반의 Cool URI 지원이 가능한 웹 컴포넌트를 설계 및 구현하였다. 본 논문에서 제안한 웹 컴포넌트는 블로그나 게시판 시스템 등에서 한글 Cool URI를 사용하고자 하는 경우, 유용하게 사용이 가능할 것으로 기대된다. 향후에는 본 연구를 바탕으로 웹 2.0과 시맨틱 웹에서의 데이터 결합 시에 다국어 식별자가 어떻게 이용될 수 있는지에 대한 연구를 진행할 예정이다.

참고문헌

[1] 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

[2] 김은수, 송강수, 이원돈, 송정길, "웹 마이닝을 이용한 개인 광고기법에 관한 연구," 컴퓨터정보학회 논문집 8권 4호, 2003

[3] Heflin, J. and Hendler, J., "A Portrait of the Semantic Web in Action," IEEE Intelligent Systems, Vol.16, No.2, pp.54-59, 2001
[4] Berners-Lee,T., Hendler, J., and Lassila,O, "The Semantic Web," Scientific American, Vol.284, No.5, pp.34-43, 2001
[5] 이우기, "하이퍼텍스트 정보 관점에서 의도적으로 왜곡된 웹 페이지의 검출에 관한 연구," 컴퓨터정보학회 논문집 10권 1호, 2005
[6] Tim, B-L.,"Cool URIs don't change," W3C style guide, 1998
[7] 김중태, "웹 2.0 시대의 기회: 시맨틱 웹", 디지털미디어 리서치, 2006
[8] Duerst, M., "Internationalized Resource Identifier," IETF RFC 3987, 2005
[9] The Unicode Consortium, "The Unicode Standard, Version 4.0," 2003
[10] Korea Industrial Standards Associations, "Hangul Unix Environment," Korean Industrial Standard, 1992
[11] Yergeau, F. "UTF-8, a transformation format of ISO 10646," IETF RFC 3629, 2003
[12] Tim, B-L. and et. al., "Uniform Resource Identifier (URI): Generic Syntax," IETF RFC 3986, 2005
[13] Fielding, R., "Hypertext Markup Language -- HTTP/1.1," IETF RFC 2616, 1999

저 자 소 개

정 의 현

1992년 한양대학교 전자공학사
1994년 한양대학교 전자공학석사
1999년 한양대학교 전자공학박사
1999년 ~ 2002년 대우통신 선임연구원
2004년 ~ 현재 : 안양대학교 디지털미디어 공학과 전임강사
관심분야 : 시맨틱 웹, 디지털 컨버전스

김 원

1984년 한양대학교 전자공학사
1989년 한양대학교 전자공학석사
2002년 경희대학교 전자공학박사
1984년 ~ 1987년 국방과학연구소 연구원
1989년 ~ 1992년 데이콤 대리
1989년 ~ 1999년 한국전산원 선임연구원
1999년 ~ 현재 : 한국인터넷 진흥원 단장

송 관 호

1980년 서울대학교 전자공학사
1984년 한양대학교 전자공학석사
1995년 광운대학교 전자공학박사
1985년 ~ 1987년 데이콤 미래연구실장
1987년 ~ 1994년 한국전산원 정보통신표준담당 연구위원
1995년 ~ 1999년 한국전산원 국가정보화센터 단장
1999년 ~ 현재 : 한국 인터넷 진흥원 원장
2005년 ~ 현재 국회과학기술정보통신위원회 정보통신 정책자문위원

박 찬 기

1994년 美 George Mason대학교 전산학 학사
1994년 ~ 1995년 선경정보시스템 사원
1995년 ~ 1999년 한국전산원 주임연구원
1999년 ~ 현재 한국인터넷진흥원 기술연구팀 팀장