

무선인터넷용 Mobile Java Browser(MJB) 개발

진민식[†] · 정민수^{**}

요 약

무선인터넷은 유선망을 중심으로 한 인터넷 비즈니스의 범위 만큼이나 다양한 형태의 데이터 제공 및 서비스가 가능해야 한다. 하지만, 무선 인터넷 초기의 SMS 기능을 이용한 단순 데이터 서비스를 위한 브라우저 만으로는 다양한 멀티미디어 콘텐츠를 서비스할 수 없으므로 좀더 발전된 브라우저가 요구된다. 본 논문에서는 이동단말기용 웹브라우저에 자바가상기계를 추가하여 자바프로그램 수행이 가능한 자바브라우저를 설계·구현하였다. 본 자바브라우저를 탑재한 이동단말기는 원하는 기능의 자바 프로그램을 동적으로 적재하여 전자상거래, 전자지갑, 게임기와 같은 다용도의 이동단말기를 가능하게 한다.

Development of Mobile Java Browser for Wireless Internet

Min-Sik Jin[†] and Min-Soo Jung^{**}

ABSTRACT

Wireless Internet has to offer various types of services as much as wired network does in the scope of Internet business. The browser for simple data service using SMS in the early wireless internet cannot serve various multimedia contents. In this paper, We design and implement a Java browser that runs Java program, by adding JVM(Java Virtual Machine) into the web browser for mobile terminals. The Java browser for mobile terminals dynamically loads various Java program you want. This browser makes it possible to be multipurpose mobile terminals for E-Commerce, E-Wallet, Game Device.

Key words: 자바가상기계, Personal Java, 브라우저, 무선인터넷

1. 서 론

무선인터넷은 사용자가 이동하면서 무선으로 음성과 데이터, 영상 등의 정보를 송·수신할 수 있는 서비스를 말한다. 초기의 무선 인터넷은 SMS방식의 단문 메시지형식의 서비스였지만, 기술 발전으로 인한 현재 무선인터넷기술은 유선인터넷에서 사용하는 인터넷 비즈니스의 범위만큼이나 다양한 형태의 데이터 제공 및 서비스가 가능하게 되었다. 그러나 이러한 무선 인터넷의 기술발전에도 불구하고 이동통신망의 낮은 대역폭과 상대적으로 작은 디스플레이 창과 같은 제약사항 때문

에 다양한 콘텐츠를 서비스할 수 있는 이동단말기용 웹브라우저 기술이 무선인터넷기술과 발맞추지 못하고 있다[16].

본 논문에서는 다양한 형태의 콘텐츠와 멀티미디어 서비스가 가능한 이동단말기용 자바브라우저를 설계하고 구현한다. 자바가상기계를 이동단말기에 탑재함으로써 자바언어측면에서 제공하는 동적 다운로드와 보안기술을 그대로 웹브라우저에 적용할 수 있고, 유선 인터넷에서 사용되는 자바 애플릿 기술을 무선 인터넷에서도 가능하게 된다.

본 논문에서 구현한 자바기계는 Personal Java(이하 PJava)를 기반으로 하였으며, 이는 메모리나 리소스가 극히 제한된 소형기기에 적합한 자바기계이다. 초기의 자바가상기계를 탑재한 소형 기기들은 자바라는 무거운 언어를 견디낼 만한 상황이 아니었고 프로그래머의 재량에 따라 약간의 성능을 높였다해도 OS

본 연구는 경남대학교 학술 논문게재 연구비 지원으로 이루어졌음

접수일 : 2002년 11월 12일, 완료일 : 2002년 12월 4일

[†] 준회원, 경남대학교 컴퓨터공학과 박사과정

^{**} 종신회원, 경남대학교 정보통신공학부 교수

차원에서 직접 운용되는 타 언어에 비해 자바기계 위에서 운용되는 자바는 느낄 수밖에 없었다. 이러한 문제점 해결을 위해 PJava가상기계는 네트워크에 연결할 수 있는 소형기에 적합한 소규모의 자바 실행 환경을 정의한 자바기계이다. 기본적으로 PJava가상기계는 JDK 1.1.8 API의 대부분의 기능을 수용하고 있으며 PDA, PPC 그리고 HPC와 같은 소형 휴대기에 적합한 자바기계이다[2,3,15,16].

이동단말기에 내장되어 PJava가상기계 위에서 동작하는 자바언어로 작성된 본 자바브라우저는 자바언어의 가장 큰 특징 중의 하나인 동적인 애플리케이션(Applet)의 다운로드를 통하여 유선에서 사용되는 애플릿 기술을 그대로 무선환경에서도 도입할 수 있게 하고. 일반적인 브라우저의 기능인 인터넷 웹 서버에 자료를 요청하여 받은 문서에 대하여 태그 중심의 렌더링(Rendering)을 하여 화면에 디스플레이 하도록 하였다.

2. 관련 연구

본 장에서는 자바브라우저를 PDA나 IMT2000기기와 같은 이동단말기에서 구동시키기 위해 필요한 자바 가상기계에 대해 알아본다. 또, 오픈 소스 브라우저인 Mozilla와 자바언어로 작성된 HotJava브라우저, C언어로 작성된 게임프로그램을 이동단말기로 다운로드 하는 기술인 GVM에 대해 살펴본다.

2.1 Personal Java

2.1.1 PJava의 개요 및 구성

PJava는 개인용 내장형 장비에서 수행되는 네트워크 응용프로그램을 위한 자바 애플리케이션 환경이다. 이는 개방형 시스템 운용기술로 기존의 데스크탑보다는 스크린 폰, PDA, 웹 TV, 디지털 셋톱박스등 소비자 정보 가진 제품에 맞는 애플리케이션에 활용되고 있다.

PJava API는 크게 선택적 패키지와 필수 패키지로 나눌 수가 있다. 필수 패키지에는 java.awt, java.applet, java.net이 있는데 이러한 패키지를 사용함으로써 인해 네트워크와 사용자 그래픽 인터페이스의 특징을 지닌 타겟디바이스에 애플릿을 실행할 수 있는 것이다. 선택적 패키지에는 java.rmi, java.sql등이 있으며 선택적 패키지의 지원여부는 타겟디바이스 설정에

맞는 PJava가상기계의 구현자 선택에 달려 있다[1,2].

2.1.2 이동단말기용 PJava

이동단말기에서 자바언어로 작성된 브라우저를 실행하기 위해서는 우선 단말기에 자바가상머신이 탑재되어 있어야 한다. 본 논문에서는 썬사에서 제공하는 WindowsNT용 PJES소스코드를 사용하여 타겟 플랫폼인 iPAQ에 PJava가상기계를 설치하였다. 타겟 플랫폼인 iPAQ PDA는 StrongARM칩과 WinCE를 OS로 탑재하고 있다.

그림 1은 PJava가상기계의 내부 구조를 나타내고 있다[2].

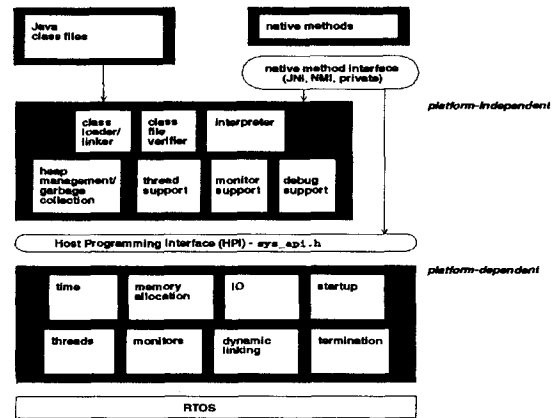


그림 1. PJava가상머신의 구조

2.2 Mozilla 브라우저

Mozilla는NGT(Next-Generation-Technology)라는 새로운 브라우저 엔진을 이용하여 보다 빠르고 효율적으로 데이터를 사용자의 모니터에 나타내게 만들 뿐 아니라 HTML 4.0, CSS(Cascading Style Sheets)1, CSS2, DOM(Document Object Model), XML (eXtensible Markup Language)등 웹 표준들을 지원하는 공개 소프트웨어이다.

Mozilla의 렌더링모듈은 그림 2와 같이 6개의 메소드를 구현하여 처리하고 있다. 생성된 트리의 Element들에 대한 위치를 정한 후 배치하는 메소드와 스크롤에 대하여 설정하는 메소드와 웹브라우저에 기본적으로 설치가 되는 Toolbar에 대한 배치를 정하는 메소드들로 구성되어 진다[4].

2.3 HotJava 브라우저

자바로 작성되어진 HotJava브라우저와 다른 브라

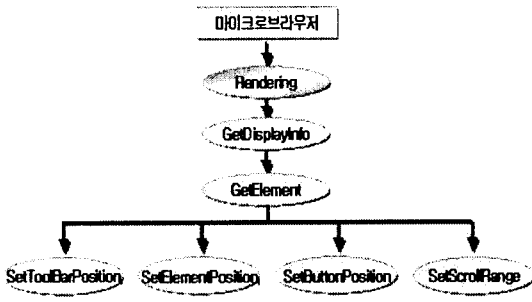


그림 2. Mozilla브라우저의 렌더링 부분의 구성

우저와의 근본적인 차이는 필요한 데이터 유형(type), 프로토콜(protocols) 및 그 사용 방법들에 대한 내용을 브라우저가 일일이 알아야 할 필요가 없다는 것이다. 즉, 콘텐츠 개발자가 각기 다른 자신의 프로토콜과 콘텐츠로 개발하더라도 자바브라우저는 프로토콜 핸들러와 콘텐츠 핸들러를 일단 자신의 파일시스템에서 찾는다. 만약, 없다면 참조할 사이트를 검색하여 해당 프로토콜의 핸들러와 콘텐츠 핸들러를 찾은 다음 자신의 로컬컴퓨터에 이 핸들러들을 다운로드하여 설치하는 방식으로 동작한다. 이러한 이유로 많은 융통성을 제공할 뿐만 아니라 새로운 능력들을 보다 쉽게 브라우저에 추가할 수 있도록 해준다. 그리고, 브라우저의 렌더링은 텍스트와 이미지들을 Java의 AwtToolkitAPI 과 Component들을 이용하여 사용자 화면에 보여주는 방식으로 처리된다[5].

2.4 GVM(General Virtual Machine)

GVM(General Virtual Machine)은 단말기용 Virtual Machine으로 사용자가 서비스에 접속하여 무선 인터넷 상에서 실행 가능한 프로그램을 다운받아 이를 단말기에서 실행 할 수 있도록 해 주는 기술이다. 서비스에 접속하여 게임이나 동영상과 같은 다채로운 멀티미디어 콘텐츠를 휴대폰으로 다운로드 하여 이용할 수 있다. 이런 서비스는 기존의 것들과는 달리 그 자체가 실행 가능한 프로그램으로 다운받은 후에 곧바로 휴대폰에서 실행이 되므로 다이나믹하고 현장감을 더욱 극대화시켜 준다. 즉, GVM에서 다운로드받은 콘텐츠는 자동으로 단말기에 저장되어 언제 어디서나 반복해서 실행할 수 있으며, 다른 콘텐츠들도 다운로드받아 갱신시킬 수 있으므로 항상 새로운 게임 등의 멀티미디어 콘텐츠를 서비스할 수 있는 기술이다[13,14].

그림 3은 GVM모듈의 다운로드 및 실행순서를 나

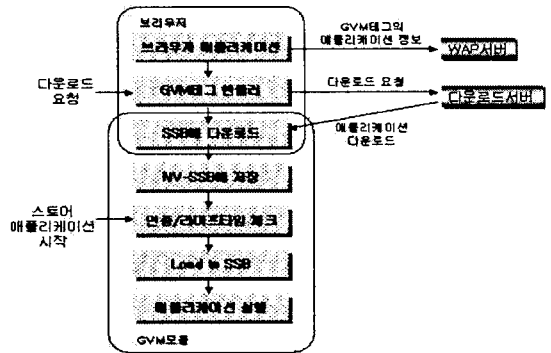


그림 3. GVM의 애플리케이션 다운로드

타내고 있다. GVM애플리케이션 서비스는 브라우저와 밀접한 관계를 가지므로, GVM애플리케이션 다운로드에는 브라우저의 GVM태그를 통해 시작된다. GVM태그에는 애플리케이션의 ID정보, 다운로드 서버 연결 정보등이 파라미터로 정의되어 있다. 사용자가 GVM태그를 선택하면, 단말기의 브라우저는 파라미터에 정의된 정보에 따라 다운로드를 시작한다[14].

GVM의 응용프로그램 다운로드기술을 HTML의 애플릿에 적용하여 무선환경에서 응용프로그램을 다운로드하고 업데이트할 수 있도록 하였다.

3. Mobile Java Browser(MJB)의 설계

본 논문에서 설계하고 구현한 Mobile Java Browser(이하 MJB)는 자바기술로 작성된 웹브라우저 컴포넌트이다. 즉, 어떤 다른 자바 응용프로그램에 포함되어 웹패드, 인터넷 스크린폰, 디지털 셋탑박스 등과 같은 인터넷기에 탑재되어 동작할 수도 있고, 본 논문에서와 같이 이동단말기에 내장되어서 standalone방식의 자바 응용프로그램으로도 동작할 수 있다.

MJB는 인터넷 상에서 동적인 콘텐츠처리를 기반으로 하는 구조로 설계되었다. 동적인 콘텐츠 처리를 위해서 Pilot이라는 개념을 사용하였으며 각각의 콘텐츠들에 대하여 동적으로 Pilot을 생성함으로써 기존 웹브라우저의 플러그-인 방식과 콘텐츠를 다루는 방식이 다르다[8,9].

3.1 Mobile Java Browser(MJB)의 구조

MJB는 필수적인 두개의 컴포넌트와 몇 개의 선택

가능한 컴포넌트로 구성된다. 필수적으로 요구되는 컴포넌트는 Pilot컴포넌트와 Scripeter컴포넌트이다. Pilots컴포넌트는 HTML/XML컨텐츠에 대한 실질적인 렌더링에 대한 책임이 있는 컴포넌트이다. 또, Scripeter컴포넌트는 지원하는 Scripting언어의 명령을 해석하고 실행하는 엔진에 대한 인터페이스 컴포넌트이다. 그 외 선택 가능한 컴포넌트는 멀티미디어를 지원하기 위한 JMF컴포넌트, Applet지원을 위한 Applet 컴포넌트, 서버와의 보안을 유지하기 위한 SSL컴포넌트들로 구성된다.

필수 컴포넌트인 Pilot과 Scripeter의 결합을 통하여 좀더 향상된 HTML/XML렌더링과 웹브라우저를 제공한다. Pilot은 각 컨텐츠마다 자바언어로 작성되며 MJB와 통합 가능하도록 설계되었다. MJB에서는 HTML과 XML을 처리하는 HTML Pilot이 필수 컴포넌트이며, 멀티미디어 데이터를 처리하는 JMF Pilot, 애플릿을 처리하는 Applet Pilot이 선택 가능한 컴포넌트이다. 그림 4는 MJB를 구성하는 Pilot과 Scripeter, 추가적인 컴포넌트를 나타내고 있다.

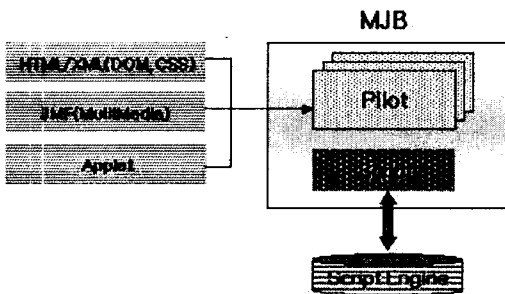


그림 4. MJB의 구조

3.2 Pilots

MJB는 여러 개의 Pilots을 하나의 레이어로 통합함으로써 동시에 많은 Pilots들을 지원할 수 있도록 하였으며, 문서에 삽입된 Applet을 서비스하기 위해 HTML Pilot이 MJB의 API를 통해서 Applet Pilot을 액세스하도록 하였다.

Pilot은 Content-Loader로부터 가져온 문서의 ContentType의 값에 따라 생성된다. 즉, 문서의 ContentType인 text/html, text/text, application/xml에 따라 다른 Pilot을 생성한다. Pilot은 Content-Loader로부터 가져온 컨텐츠를 파싱하고 Viewport를 통해 디스플레이 하는 객체이다. 또, MJB 사용자가

Stop버튼을 누르게 되면 현재 브라우징 작업을 멈추는 기능과 화면 확대/축소와 같은 기능을 처리하도록 하였다.

그림 5는 MJB가 HTML, XML과 같은 웹 문서를 가져와서 Pilot을 생성하는 과정을 보여주고 있다. MJB사용자의 HTTP요청에 대한 응답헤더의 Content-Type의 값에 따라 Pilot객체를 생성한다.

그림 6은 로컬시스템에 있는 Pilot생성을 위한 파일의 예를 보여주고 있다. 왼쪽은 MIME타입과 파일확장자를 나타내고 오른쪽은 MIME타입에 따라서 생성되는 Pilot의 경로와 지원하는 컨텐트타입을 나타낸다.

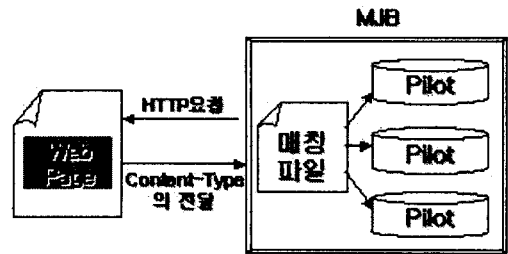


그림 5. 매칭 파일을 통한 Pilot생성과정

<pre> application/vnd.wap.wmlc: description=WAP WML binary: file_extensions=.wmlc text/vnd.wap.wml: description=WAP WML text: file_extensions=.wml text: description=Pure Ascii Text: file_extensions=.txt </pre>	<pre> pilot name=HTML 4 Pilot: description=HTML 4 pilot: classname=ice.pilots.html.ThePilot: content_types=text.html, text.xml, application.xml, content.unknown pilot: name=Text Pilot: description=Ascii text pilot: classname=ice.pilots.text.ThePilot: content_types=text.text, text.plain </pre>
---	--

그림 6. 매칭 파일의 예

3.3 HTML Pilot

HTML Pilot은 MJB의 핵심 컴포넌트로서 기능별 모듈들로 구성되어 있다. 웹서버로부터 전달된 HTML/XML문서의 렌더링에 대한 책임이 있다. 그림 7에서와 같이 각 모듈들은 객체지향개념을 기반으로 설계되었고 각 모듈은 전체에 영향을 주지 않으면서 어느 한 부분을 수정하거나 대체할 수 있도록 구현되었다.

3.3.1 Content-Loader

Content-Loader는 HTML/XML파서 모듈에 포함

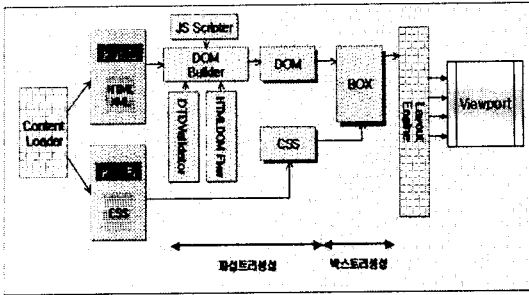


그림 7. 렌더링 모듈의 구성요소

되어 있고, 웹 상에 있는 콘텐츠를 가져오는 부분이다. 본 논문에서는 표 1과 같이 여러 가지 방법으로 콘텐츠를 가져올 수 있도록 하였다. 아래의 표 1은 다양한 방법으로 콘텐츠를 가져오기 위한 Content-Loader의 생성자를 나타내고 있다.

표 1. Content-Loader의 생성자

Constructor	description
ContentLoader(java.net.URLConnection urlc, java.lang.String contentType)	URLConnection을 통해
ContentLoader(java.net.URL base, java.io.InputStream in, java.lang.String contentType)	InputStream을 통해
ContentLoader(java.net.URL u, java.lang.String contentType)	URL을 통해
ContentLoader(java.net.URL base, java.lang.String loc, java.lang.String contentType)	base Location인 URL에서 상대적인 위치인 loc의 위치

3.3.2 HTML/XML 파서와 CSS파서

HTML/XML 파서는 HTML과 XML의 입력을 처리하는 역할을 담당한다. 문서 안의 모든 태그들에 대하여 포함관계와 형제관계를 나타내는 Data Tree(문서 계층 구조)를 생성하고 DOMBuilder객체와 상호 작용하는 논리적인 컴포넌트이다.

CSS 파서는 브라우저 자체나 웹 페이지로 다운로드된 CSS파일(StyleSheet파일)을 파싱하여 CSSDecl 객체와 CSSRule객체를 생성하는 역할을 한다. 그리고, HTML/XML 파서에서 웹페이지 파싱 중 나타나는 <style>..</style>태그 사이에 있는 스타일 명령들도 CSSDecl객체와 CSSRule객체를 생성한다[6].

3.3.3 DOMBuilder

이 모듈은 Content-Loader로 부터 가져온 콘텐츠를 파서를 통해 DOM트리를 만드는 모듈이다.

그림 8은 DOMBuilder객체가 콘텐츠를 파싱하여

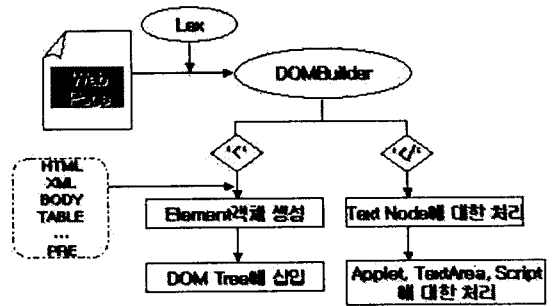


그림 8. DomBuilder를 통한 Element생성과정

DOM트리를 생성하는 과정을 도식화한 것이다. 트리 생성과정은 전달된 문서를 Lex를 통해서 태그 중심으로 파싱하고 DOMBuilder가 품과 관련된 태그인지, 프레임과 관련된 태그인지, 이미지관련 태그인지에 따라서 다른 Element를 생성한다. 그리고, 생성된 Element를 DOM트리에 삽입한 후 끝 태그를 만나게 되면 하나의 Element에 대한 처리를 마친다. 만약 태그 뒤에 텍스트가 있으면 TextNode를 삽입한다. 이처럼 DOMBuilder는 하나의 문서에 있는 모든 태그들에 대하여 Element를 생성한다.

3.3.4 DTDValidator/HtmlDOMFixer

현재 모든 HTML문서는 문서의 구조를 알지 못하므로 invalid하게 제작되어 진다. 즉, 문서의 DTD를 가지고 있지 않다. 그리고, 상용 브라우저는 정확한 DTD를 가지지 않는 invalid한 HTML문서에 대하여도 화면에 디스플레이하고 있다.

하지만, MJB는 DTDValidator객체를 통해서 DTD를 가지는 XML문서에 대하여 DTD의 유효성을 검증하고, invalid하게 제작된 HTML문서는 HTMLDOMFixer객체를 통하여 보정한다. 이 모듈은 Document Type Definition(DTD)를 포함하고 있다. DOMBuilder가 DOM트리 안으로 정확하지 않은 노드를 삽입하려고 할 때 호출된다. DOMFixer는 DOM트리를 검색하면서 DTD에 따라 올바른 위치를 찾아 노드를 삽입한다. 예로써, HTML문서 파싱 중 <table>태그가 없는 <tr>태그를 만나게 되면 <table>태그를 적당한 곳에 삽입 한 후 <tr>태그를 자식으로 등록하도록 하는 역할을 한다.

3.3.5 BOX

DOM트리의 각 Element에 스타일을 적용하여 화면

상에 디스플레이하기 위해 본 논문에서는 BOX라는 새로운 개념을 도입하였다. BOX는 DOM트리의 각 Element에 CSS를 적용하여 화면상에 배치하기 위한 사각형의 박스라고 할 수 있다. 이러한 Element에 대한 CSS 명령은 각 Element마다 관리되어 져야 하며 BOX를 만들 때 필요하게 된다.

이러한 BOX의 개념을 도입하기 위해 W3C의 CSS2 일부를 자바언어로 구현하였다. BlockBox, InlineBox, ButtonBox와 같은 BOX객체를 CSS2규칙에 맞게 적용하였다.

3.3.6 CSS와 Layout Engine

CSS는 그림 7에서와 같이 Cascading Style Sheet (CSS)를 파싱한 후, 각 노드별 CSS객체를 생성한다. 즉, DOM트리에 있는 각 노드에 대한 스타일 명령을 저장하고 있는 역할을 한다.

Layout Engine은 DOM트리의 각 노드에 CSS객체를 적용하여 스크린상에 각 노드를 보여주는 역할을 한다.

3.3.7 Viewport

MJB는 복잡한 웹 페이지를 디스플레이 하기 위해 Viewport개념을 사용하였다. Viewport는 Pilot을 위한 컨테이너로써 역할을 수행한다. 이러한 Viewport는 HTML 렌더링 모듈과 분리되어 있다. 이것은 문서에 삽입된 Applet을 동일한 Viewport에 보여주기 위함이다. 즉, Pilot이 변하더라도 같은 Viewport에서 다양한 콘텐츠를 디스플레이 할 수 있는 기능을 제공한다.

3.4 Scripter

본 논문에서는 웹페이지안에 삽입되어 있는 자바스크립트 명령을 지원한다. Scripter는 <script>태그 안에 들어 있는 자바스크립트 명령을 Rhino엔진을 통해 해석하고 실행한다. 즉, MJB는 Rhino엔진에 대한 인터페이스만을 가지고서 스크립트 명령을 처리 할 수 있다. 다른 스크립트 명령도 자바스크립트와 마찬가지로 스크립트엔진에 대한 인터페이스만을 구현함으로써 쉽게 통합되어 질 수 있다[4].

4. Mobile Java Browser(MJB)의 구현

본 논문에서 구현한 MJB는 웹 상에 존재하는 컨텐

츠를 다운로드하고 DOMBuilder객체를 이용하여 DOM트리를 만든 후, DOM트리의 각 노드들에 CSS 값을 적용하여 BOX트리를 생성하는 부분과 생성된 BOX들을 Viewport에 디스플레이 하는 부분으로 크게 나눌 수 있다. BOX트리를 만들기 위해 DOMBuilder, HTMLDOMFixer, CSSMatcher, CSSAttribs객체들이 사용되며, Viewport에 디스플레이를 위해 Layout Engine과 이벤트를 처리하는 DocPanel객체가 사용된다. 그리고, 이 장의 마지막에는 이동단말기에 탑재되는 PJava가상기계와 MJB의 소스코드구성이 있다.

4.1 DOMBuilder객체의 구현

DOMBuilder객체는 DOM트리를 생성하는 모듈이다. DOMBuilder는 HTML/XML파서를 가지고 있으며 Valid한 DOM을 생성하기 위한 검증자 역할까지도 한다. 하나의 문서에 대한 DOMBuilder의 수행과정은 사용자가 입력한 URL에 있는 웹 문서를 읽어와서 문자배열에 저장하고 문자배열에서 하나의 문자씩 읽어와 < 와 > 와 짝이 되는 >를 만나게 되면 Element생성을 시작하게 된다. Element의 생성은 문자배열에 해당하는 태그의 ID를 우선 구한 다음 ID를 매개변수로 하여 Element를 생성한다.

아래의 표 2에서와 같이 여러 종류의 Element들이 있다. 또, 특별한 태그인 APPLET, FORM에 관련된 태그에 대해서는 여러 가지 속성값들을 저장할 수 있는 특별한 Element를 생성한다.

표 2. 생성되는 Element객체들

Tag	Element	Tag	Element
A	DAnchorElement	Select	DSelectElement
APPLET	DAppletElement	Option	DOptionElement
AREA	DAreaElement	Input	DInputElement
Form	DFormElement	Map	DMapElement
Iframe	DIFrameElement	TextArea	DTextAreaElement
Img	DImageElement	텍스트	DTextNode

표 2에 있는 모든 Element는 Element 인터페이스로부터 구현된 클래스이다. 그림 9는 Element interface에 대한 DAnchor와 DSelect Element에 대한 계층구조를 나타내고 있다.

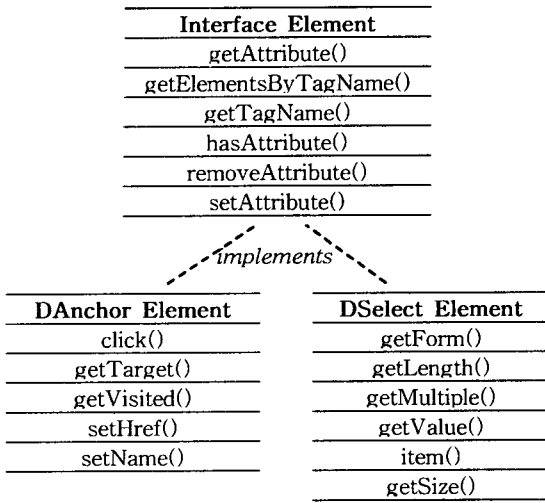


그림 9. DAnchor, DSelect Element의 계층관계

4.2 HtmlDOMFixer객체

MJB는 HtmlDOMFixer객체를 이용하여 invalid한 HTML문서에 대하여 보정을 하면서 DOM트리를 만든다. 이 모듈은 HTML을 위한 DTD를 포함하고 있으면 DTD의 검증을 수행하고 퍼지 이론을 이용한 Fixer 객체가 정확한 HTMLDTD를 생성한다. 이러한 Fixer 객체는 DOMBuilder객체가 DOM트리 안으로 정확하지 않은 Element를 삽입하려고 할 때 호출되어지고 정확한 위치에 노드를 삽입하기 위해 DOM트리를 여행하게 된다.

```

<TR>
  <TD>테이블1</TD>
</TR>
</TABLE>
```

위의 예제와 같이 <table> 태그가 없는 <tr>태그를 만나게 되면 파서는 우선 <tr>태그를 적당한 곳에 삽입하려고 시도하지만 DOM은 <tr>노드를 삽입하기 위해서 <table>태그를 요구하고 있다. 그래서 DOMFixer는 시작 <table>태그를 찾기 위해 DOM트리를 찾아가게 된다. 그러나, DOM트리의 어느 곳에서도 <table>태그는 없으므로 DOMFixer객체는 합법적인 <tr>의 위치를 정하기 위해 적당한 곳에 우선 <table> 태그를 삽입하게 된다.

4.3 BOX객체

만들어진 DOM트리를 Viewport에 보여주기 위해

Element마다 BOX객체를 생성한다. BOX객체는 적용될 스타일명령인 CSSAttribs에 저장되어 있는 CSS값을 CSSMatcher를 이용하여 찾는다. 그림 10은 BOX객체를 생성하는데 사용되는 여러 객체들과 BOX객체 생성과정을 보여주고 있다.

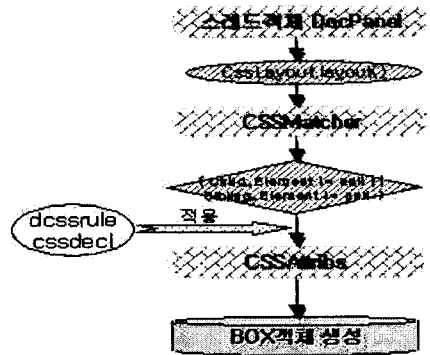
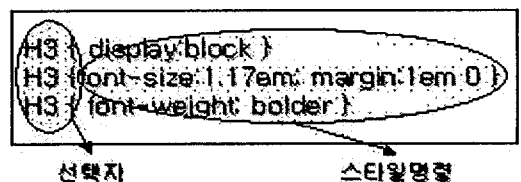


그림 10. BOX 생성 과정

그림 10에서와 같이 CSSMatcher는 문서의 Element에 대하여 어떤 스타일을 적용해야 하는지를 선택하는 객체라고 할 수 있다. 즉, 문서 트리의 Element에 어떤 스타일 패턴을 적용해야 하는지를 판별한다. 선택자라 불리는 이들 패턴은 단일 패턴에서부터 훨씬 더 복잡하고 풍부한 패턴들이 될 수 있다. 만일 패턴 안의 모든 조건들이 어떤 Element에 맞다면, 그 선택자는 그 Element와 일치한다고 할 수 있다. 이렇게 CSS로부터 문서의 Element와 일치하는 선택자를 선택하는 역할을 하는 것이 CSSMatcher이다.

CSSAttribs는 문서의 Element들에 대하여 각각 하나씩 존재한다. Element마다 생성되는 CSSAttribs는 CSS에 나온 속성의 전달과 비전달을 정확하게 구현하기 위해 부모의 속성을 매개변수로 전달하여 CSSAttribs를 생성한다. CSSAttribs는 CSS문법을 지원하기 위해 각 Element에 적용이 되어질 수 있는 속성들에 대한 모든 정보들을 클래스 내부에 포함할 수 있도록 구현하였다.



위의 스타일 명령은 CSS파일에 있는 H3에 관련된 속성들이다. 즉, 웹문서에서 H3태그가 나타나면 위의 3개의 스타일 명령이 적용되어 디스플레이 되어야 한다는 것이다. H3태그는 디스플레이가 블록(block)이고 FONT의 크기가 1.17em이고 margin-top과 margin-bottom이 1em이고 margin-left, margin-right값이 0으로 설정되고 FONT굵기가 bolder값을 가지도록 화면에 보여야 한다는 것이다.

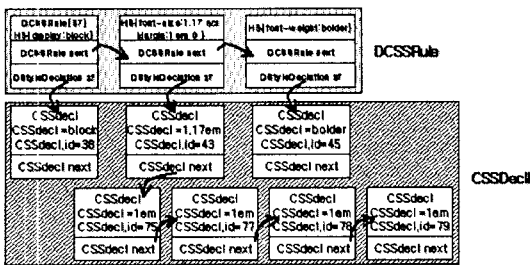


그림 11. 생성된 DCSSRule과 CSSDecl객체들

그림 11는 위의 예제에 대하여 CSS파일을 파싱을 한 후 생성되는 DCSSRule과 CSSDecl의 자료구조를 나타내고 있다. 태그마다 생성되는 DCSSRule과 CSSDecl의 자료구조를 가지고서 CSSAttribs의 속성값을 변경시키고 이 CSSAttribs속성값을 가지고서 스타일이 적용된 BOX를 생성할 수 있다.

4.4 BOX의 생성

각 Element에 대하여 생성되는 BOX는 CSSAttribs의 display, position속성값에 따라 여러 종류의 BOX들을 생성한다. 그리고 BOX종류에 따라서 화면에 보여주는 방식에 차이가 있다. BlockBox는 자식으로 화면에 보이지 않는 박스를 가지고 있다. 즉, 다른 BlockBox를 포함하고 있는 BOX있다. InlineBox는 TextBox를 자식으로 가지고 있거나 이미지태그에 의해 생성된 BOX처럼 화면에 보이는 BOX이다.

아래는 CSSAttribs의 display, position의 값에 따라 생성되는 BOX의 종류를 나타내고 있다.

- BlockBox: display특성이 block의 값을 가지는 Element에 대하여 생성되는 박스이다. 하위(descendant)박스의 내용을 위한 용기 블럭을 생성한다.
- ObjectBox: Img, Applet, Form관련 Element에 대하여 생성되는 박스이다. 외부URL에 있는 이미지 자원을 가져오거나 Applet의 처리를 위한 Applet

Viewer클래스를 생성하여 Applet을 관리한다.

- BreakBox: Br태그에 대하여 생성되는 박스로써 크기가 1인 BlockBox의 역할을 한다.
- InlineBox: 실질적으로 화면에 보여주는 박스이고 Inline박스는 문단을 구분하지 않고 오른쪽으로 계속해서 배치된다.
- BulletBox: Element에 의해 생성되는 박스로서 리스트를 앞쪽 문자로 처리한다.
- FloatBox: 현재의 줄에서 오른쪽 또는 왼쪽으로 변환되는 하나의 박스이다.
- TableBox: <Table>Element에 의해 생성되는 박스이다.
- TextBox: TextNode에 의해 생성되는 박스이다.
- FormBox: Button, Choice, Select, Checkbox, Textfield에 의해 생성되는 박스이다.
- AppletBox: Applet Element에 의해 생성되는 박스이다.

위와 같은 방법으로 Top-Element[HTML]부터 단 말 노드까지 모든 Element의 display, position값에 따라 적당한 박스객체를 차례대로 만든다. 여기서도 DOMBuilder에서 만든 파싱트리와 같이 박스트리를 생성한다. 이렇게 생성된 박스트리도 자식-부모관계를 가지는 계층적인 모양을 하고 있다. 이렇게 모든 Element에 대하여 박스를 모두 만들고 나면 화면에 박스객체들을 배치하도록 구현하였다.

4.5 Layout Engine의 구현

Layout Engine은 DOM으로부터 BOX를 만든 다음 스크린으로 Element를 표현하는 모듈이다. Layout Engine은 DOM과 CSS를 가지고서 가장 빠르고 효율적인 랜더링을 위한 책임이 있다.

본 논문에서 구현한 MJB의 컴포넌트 랜더링은 더블 버퍼링을 이용한 고성능의 랜더링이 가능하며 AWT와 Swing컴포넌트를 지원한다. 또, 동적인 콘텐츠 핸들링을 통한 3D그래픽의 기능도 가능하다. 더블 버퍼링은 이중버퍼를 사용하여 화면에 빠르게 복사하는 방식으로 화면의 떨림 현상을 방지하였으며 사용자가 보기에는 화면이 훨씬 더 부드럽게 갱신되는 것처럼 보이게 된다.

4.6 이벤트 처리 - DocPanel

모든 GUI프로그램은 이벤트 드리븐(Event-Driven)

방식으로 동작한다. 즉, 정해진 프로그램의 흐름이 있는 것이 아니라 발생한 이벤트를 처리하는 것이 프로그램 논리의 핵심이 된다. MJB에서도 자바의 AWT1.1 이벤트 모델을 사용하여 모든 이벤트에 대한 처리를 한다. 즉, 위임형 이벤트 모델방식을 사용하는 것이다. 위임형 이벤트 모델이란 버튼이 눌러졌을 경우 이벤트 처리를 버튼에서 하지 않고 버튼에 등록된 이벤트 리스너, 즉 버튼의 이벤트를 기다리는 객체에게 이벤트 처리를 위임하는 방식이다[10].

MJB는 화면상의 사용자의 입력이라든지 DOM에서 발생하는 노드의 삽입, 삭제와 갱신에 대한 이벤트를 처리하는 루틴을 가지고 있다. 화면에 랜더링을 하는 실질적인 객체인 DocPanel에서는 사용자의 이벤트를 듣기 위해 Listener인터페이스를 구현하였으며 이렇게 발생한 이벤트에 대해서는 DOMEventAdapter 객체가 발생한 이벤트에 대한 처리를 한다. 그림 12는 DocPanel객체가 발생하는 모든 이벤트를 처리하기 위해 리스너를 등록하는 부분을 나타낸다.

```
public class DocPanel extends Panel
implements EventListener, DocumentView, AdjustmentListener, MouseListener, MouseMotionListener, CSSLayoutListener
```

DocPanel은 Panel을 상속하였으며 EventListener, AdjustmentListener, MouseListener, MouseMotionListener를 구현하였으므로 모든 키보드나 마우스에 대한 이벤트를 감지할 수 있다.

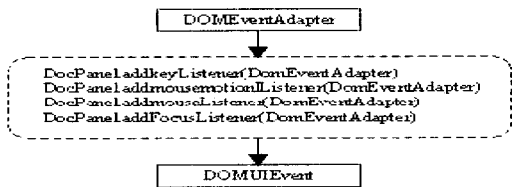


그림 12. 이벤트처리를 위한 리스너 등록

4.7 Scripeter의 구현

Scripeter는 문서 안에 포함된 스크립트 명령을 처리하는 프로그램이다. 본 논문에서 구현한 스크립트언어인 Rhino JS Engine에 의해 처리되는 자바스크립트언어이다. 즉, HTMLPilot으로부터 자바스크립트명령을 Rhino JS Engine으로 전달하는 역할을 한다. Rhino JS Engine은 스크립트를 실행 후 결과값을 Element형식으로 HTMLPilot으로 전달하다. 이렇게 전달된

Element는 DOM트리에 삽입되는 것이다.

MJB는 객체지향언어인 자바로 구현되었으므로 Python과 같은 다른 스크립트 명령도 자바스크립트와 마찬가지로 스크립트엔진에 대한 인터페이스만을 구현함으로써 쉽게 MJB와 통합되어 질 수 있다.

4.8 소스코드의 구성 및 실행화면

아래의 표 3는 PJava가상기계의 모듈별 구성을 나타내고 있다. PJava가상기계는 크게 플랫폼 독립적인 부분과 플랫폼 의존적인부분으로 구성되어 있으며 플랫폼 독립적인 부분은 썬의 오픈 소스를 그대로 사용하였으며 플랫폼 의존적인 부분은 개발대상 제품인 IPAQ PDA에 맞게 재작성 되었다[2].

표 3. PJava가상기계의 구성

Dependent부분
Class Loader/Linker
Class File Verifier
Interpreter
Heap/Garbage Collection
Thread Support
Independent부분(제작성원 부분)
Time Support Functions
Memory Allocation Functions
I/O Support Functions
Thread Support Functions
Dynamic Linking Functions

표 4는 PJava가상기계의 핵심 API의 구성과 크기를 나타내고 있다. MJB는 아래에 있는 API를 이용하여 구현되었다. AWT, NET, IO, EVENT와 같은 API들이 존재한다.

표 4. Core API의 구성 및 크기

java.applet	4.9k
java.awt	327k
java.awt.datatransfer	8.1k
java.awt.event	37.2k
java.awt.image	39.3k
java.beans	65.6k
java.io	176k
java.lang	180k
java.lang.reflect	16.6k
java.math	10.2k
java.rmi 관련	57.9k
java.security 관련	86.9k
java.text	673k
java.util	120k
java.util.zip	46.6k
java.net	80.2k

표 5은 PJava가상기계 위에 탑재되는 MJB의 모듈별 크기를 나타낸다. 각 모듈은 모두 Jar파일의 형태로 제공되어진다.

표 5. MJB의 구성 및 크기

Jar파일	Size	Description
core.jar	423K	MJB의 Core와 HTML Pilot이 포함되어 있다.
extra.jar	36K	애플릿, 멀티미디어, PDF Pilot을 포함하며, 멀티미디어를 위해 JMF2.0, PDF를 위해 Adobe Acrobat Viewer가 가 요구된다.
http.jar	183K	HTTP 프로토콜을 제공하기 위해 부분이다.
js.jar	243K	자바스크립트를 지원하기 위해 Rhino JavaScript Engine에 대한 인터페이스를 구현한 부분이다.
util.jar	37K	MJB에 필요한 유틸리티를 제공하는 부분이다.
rhino_src.jar	375K	Rhino Engine이 포함된 부분이다.

그림 13은 MJB브라우저의 실행화면이다. 다른 상용트라우저에 비해 인터페이스면에서 미흡하다. 히스토리리를 관리하는 앞/뒤로 버튼과 URL입력창, 상태표시줄이 있다.

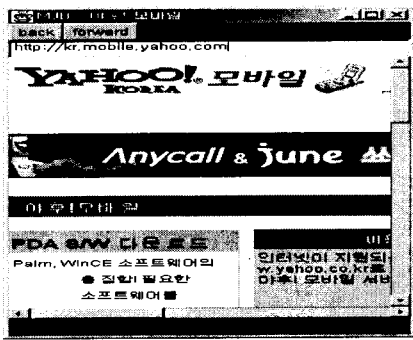


그림 13. MJB브라우저 실행화면

5. 성능 평가

본 논문에서 구현된 MJB는 StrongARM프로세스에 WindowCE3.0을 탑재하고 있는 IPAQ PDA에서 설계되고 구현되었다. 우선, 위에서 데스크탑을 위해 배포된 PJava가상머신은 X86과 WindowNT위에서 동작하는 프로그램이므로 ipaqPDA위에 PJava를 탑재하기 위해서 X86어셈블리를 Armasm으로 변경하고 임베디드 툴을 이용하여 변경하였다[11].

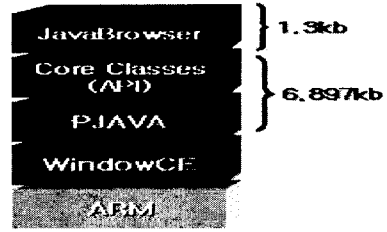


그림 14. 플래폼과 MJB

그림 14는 MJB시스템의 전반적인 구조를 나타내고 있다. 오른쪽은 PJava가상머신과 MJB의 메모리용량을 나타낸다. 테스트를 위해 디버깅모드로 컴파일되었고 PJava가상머신은 데스크탑에서 사용되는 JVM과는 달리 시스템클래스와 API들을 ROM에 미리 탑재하는 방식으로 동작한다. 그래서, 그림 14에서와 같이 조금 많은 초기 메모리를 요구하지만 릴리즈 모드 시에는 PJAVA와 API의 크기가 줄어들 것이다. 또, MJB는 시스템클래스와 API들을 미리 ROM에 탑재하는 방식(ROMIZING)을 사용하므로 파일시스템을 사용하여 필요할 때마다 ROM으로 옮기는 JVM의 방식에 비해 수행 속도면에서 성능개선이 되었다. 또, MJB는 C나 C++로 작성된 상용 브라우저에 비해 자바언어에서 지원하는 더블버퍼링과 스레드를 통한 고성능 렌더링이 가능하므로 향상된 디스플레이를 제공한다.

표 6은 현재 이동단말기에 탑재되어 있는 상용브라우저와의 차이를 도표로 나타내고 있다.

표 6. 상용 브라우저와의 비교

	WAP브라우저	MME브라우저	MJB
OS	Palm과 Window기반	Window기반	WindowCE기반
확장성	WAP프로토콜을 모두 구현해야함		GOPI기반의 구현으로 확장 용이
사용언어	WML사용	HTML서브셋사용	HTML, XML 지원
보안기능	TLS를 사용한 보안	SSL을 통한 보안	Java2보안 메커니즘의 사용
서버와의연결	WAP게이트웨이 필요	게이트웨이 필요없음	게이트웨이 필요없음
WEB문서의 보정	지원하지 않음	지원하지 않음	HTMLDomFixer를 통해서 보정
Applet지원	지원하지 않음	지원하지 않음	지원
자바 응용프로그램의 실행	실행 안됨	실행 안됨	실행 가능

6. 결 론

무선 인터넷 시장에서 주도권의 향배를 결정하는

것은 이동단말기용 웹브라우저이다. 일반적인 PC에서 인터넷을 사용하기 위해서는 넷스케이프나 익스플로러와 같은 웹브라우저 소프트웨어가 필요한 것처럼 이동전화나 PDA에서 인터넷을 사용하기 위해서는 전용 웹브라우저가 있어야 한다.

본 논문에서는 이동단말기에서 무선인터넷을 사용하기 위한 웹브라우저인 MJB를 설계·구현하였고 자바상기계를 이동단말기에 탑재하였다. 자바언어로 작성된 MJB는 자바상기계가 탑재되어 있는 모든 이동기기에 탑재가 될 수 있으며, 애플릿 기술을 이용하여 게임과 같은 멀티미디어 콘텐츠들을 다운로드받을 수 있다. 또 이동단말기에 탑재되어 있는 자바상기계는 일반 자바 응용프로그램을 실행할 수 있다.

현재 국내의 이동 단말기에 탑재되어 출시되는 대부분의 상용 브라우저는 브라우저를 개발한 외국 회사에 개당 몇 달러씩의 로열티를 지불하고 있으므로 MJB처럼 국내 기술로 개발된 브라우저는 기술적 측면과 경제적 측면에서 많은 도움이 될 것으로 기대된다. 또, 이동단말기에 자바상머신이 탑재되어 있다면 자바로 작성된 응용프로그램을 실행할 수 있는 환경을 제공할 수 있으므로 데스크탑기반의 PJava를 ARM 플랫폼으로 변경하여 탑재하는 것만으로 많은 이점을 제공할 수 있다.

참 고 문 헌

- [1] <http://java.sun.com/PJava>, *Personal Java Application Environment Specification Version 1.1.2*, Sun MicroSystem, 2000.
- [2] <http://java.sun.com/PJava>, *Personal Java Application Environment Porting Guide Version 3.1*, Sun MicroSystem, 2000.
- [3] Bill Venners, *Inside the Java Virtual Machine*, McGraw-Hill, 1998.
- [4] William R. Stanek, *Nescape Mozilla Source Code Guide*, Nescape Press, 1998.
- [5] Larry Ayers, *Running Sun's Hotjava Browser on a Linux System*, <http://linuxgazette.fst.or.kr/issue11/hotjava.html>, 1996.
- [6] Dan Livingston , Micah Brown, *essential CSS & DHTML for Web Professionals*, Prentice Hall, 1999.
- [7] Li Gong, *Inside Java(TM) 2 Platform Security: Architecture, API Design, and Implementation*, Sun MicroSystem, 2000.
- [8] Buyukkokten, Orkut ., *Power browser: efficient Web browsing for PDAs*, CONF HUM FACT COMPUT SYST PROC. 2000.
- [9] Uehara, Tomokazu, Tadao, Masahito, *AR-browser: Active Reading for WWW contents*, IEEE SYMP VISUAL LANG PROC, 2000.
- [10] Ronald Ashri, Steve Atkinson, Danny Ayers, *Java Mobile Programming*, WROX, 2001.
- [11] 김진호, *입문자를 위한 WindowCE Programming*, 가남사, 2002.
- [12] Perez Diaz, Jesus Arturo, *A secure Java2-based mobile agent system with a comprehensive security*, Informatica (Ljubljana) [Inf] vol. 24, no. 4 pp., 475-485, 2001.
- [13] HWU, YKH, KTJ, *MAP Programming Guide*, 모빌탑, 2001.
- [14] 윤종현, *모바일C 기반의 베헤얼머신 GVM 모바일 C로 속도와 안정성 실현*, 마이크로소프트웨어, pp225-234, 2001. 7.
- [15] 이상운, 이미영, 김명준, *국내 이동단말기용 VM의 개발 동향*, 콘텐츠아카데미연구원, 2001.2.
- [16] 김상덕, 2세대 무선 인터넷 세상을 여는 VM무선 인터넷, 힘찬 도약의 전주곡 베헤얼 머신, 마이크로소프트웨어, pp214-224, 2001. 7.
- [17] A. Taivalsaari, *Implementation a Java Virtual Machine in the java programming Language*, SUN Lab, 1997.
- [18] <http://www.artima.com/>, ARTIMA SOFTWARE COMPANY.
- [19] Radhakrishnan, Ramesh, *Microarchitectural techniques to enable efficient Java execution*, The University of Texas at Austin, 2000.
- [20] 정민수 이종동, *Design of High-Performance Java Virtual Machine*, 研究論文集, Vol.16 No.1, 1998.



진 민 식

2000년 2월 경남대학교 컴퓨터공학과 학사
2002년 2월 경남대학교 컴퓨터공학과 홈네트워크 전공 공학석사
2002년 3월~현재 경남대학교 컴퓨터공학과 홈네트워

크 전공 박사과정

관심분야 : JavaMachine, HomeNetworking, JavaCard



정 민 수

1986년 서울대학교 컴퓨터공학과 학사
1988년 한국과학기술원 전산학과 공학석사
1994년 한국과학기술원 전산학과 공학박사
1990~현재 경남대학교 컴퓨터공

학과 부교수

관심분야 : Embedded System, JavaMachine, Compiler

교 신 저 자

진 민 식 631-700 마산시 월영동 449번지 경남대학교 컴퓨터공학과 경남대학교 컴퓨터공학과 박사과정