전자 상거래 플랫폼 개발을 위한 멀티미디어 컴포넌트 설계 기법

김철진 , 김수동
숭실대학교 전자계산학과

Multimedia Component Design Techniques for Electronic Commerce Platform

Chul Jin Kim , Soo Dong Kim
Department of Computer Science , Soong Sil University

Abstract

현재 인터넷 상에서 급속하게 전자 상거래가 이루어지고 있으나, 아직까지 신용카드에 의한 초보적인 온라인 거래에 불과할 뿐, 본격적인 전자상거래가 이루어 지지 않고 있다. 그러나 기반 관련 기술과 법규가 표준으로 채택된다면, 빠른 시간 안에 전자 상거래의 기술적 진보가 이루어 질 수 있으며, 또한 관련 어플리케이션도 무수히 많이 개발될 것으로 예상된다. 이런 전자 상거래 관련 어플리케이션의 개발에 있어서 경쟁력은 빠르고도 범용적인 개발을 할 수 있는 플랫폼을 개발하는 것이다. 전자 상거래 플랫폼은 어플리케이션의 개발 기간을 단축하여 Time-To-Market 을 줄이며, 개발 비용의 감소와 상품 및 전자상거래 서비스를 단기간에 제공할 수 있다. 범용적인 플랫폼의 개발에 있어서 인터넷의 성능을 크게 좌우할 수 있는 것은 대량의 멀티미디어 데이터를 어떻게 빠르게 전송할 수 있는가를 설계하는 것이며, 또한 전송된 멀티미디어 데이터를 각각의 클라이언트에서 효율적으로 네트워크 트래픽(Network Traffic) 없이 운용할 수 있는 설계 기법을 제시한다.

1. 서론

전자 상거래는 인터넷 상에서 이루어 지므로 사용자의 요구에 대한 관련 어플리케이션과 그와 관련된 멀티미디어 데이터를 네트워크를 통해 원거리에 있는 사용자 호스트로 전송하므로 멀티미디어 데이터의 크기와 형태에 따라 어플리케이션의 성능을 크게 좌우한다.

따라서 객체지향 패러다임(Object-Oriented Paradigm)을 이용해 멀티미디어 컴포넌트의 설계를 효율화할 수 있는 방안을 제시하여 기존의 절차적 클라이언트/서버 모형이 지니고 있는 문제점이나 제약 사항들을 해결할 수 있으며, 성능, 효율성 및 재사용성 측면에서 웹 어플리케이션의 품질 향상에 크게 기여할 수 있다.[1][4][5]

본 논문에서는 멀티미디어 컴포넌트의 효율적인 설계를 위한 두 가지 기법을 제시한다. 멀티미디어 객체를 상품 객체와 어떻게 적용하느냐에 따라 클라이언트에서 서버로 전송되는 멀티미디어 컴포넌트의 성능이 좌우된다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구로서 기존의 웹 어플리케이션에서의 멀티미디어 데이터 표현 형태를 설명하고 단점을 제시하며, 3 장에서는 두 가지 멀티미디어 설계 기법 중 멀티미디어 클래스와 상품 클래스를 통합하여 복합 객체를 만든 구조를 제시하고, 4 장에서는 멀티미디어 클래스와 상품 클래스를 분리하여 멀티미디어 서비스를 제공하는 형태를 제시한다. 5 장에서 실험 및 평가를 하고, 마지막으로 6 장에서 결론 및 향후 연구 계획을 제시한다.

2. 기존의 멀티미디어 데이터의 표현 형태

인터넷 기반의 웹 어플리케이션을 개발하는 데 있어서 그림 1과 같은 형태를 취해오고 있다. 즉, 사용자 요청에 대한 결과는 Web Server 로부터 HTML 문서 형태로 생성되며, CGI 프로그램을 통하여 계산 및 데이터베이스 인터페이스를 제공하는 형태이다.

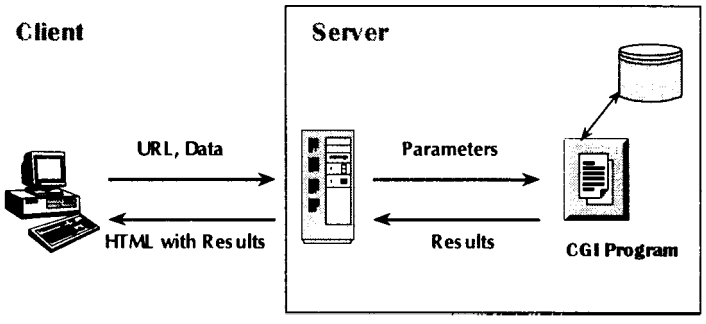

그림 1. CGI 기반의 웹 어플리케이션 구조

그림 1과 같은 방식의 웹 어플리케이션은 많은 문제점과 제약사항을 지니고 있다. 클라이언트 컴퓨터는 모든 정보의 검색과 처리를 서버에 요청해야 하고, 서버는 CGI 프로그램을 실행하여 그 결과를 클라이언트 컴퓨터에 보내게 되고 결과는

클라이언트 컴퓨터의 웹 브라우저를 통해 출력되게 된다.[2] 이 때 대량의 멀티미디어 데이터를 클라이언트가 요구 시 서버로부터 클라이언트로 전송하는데 있어서 매 요구 마다 대량의 멀티미디어 데이터를 클라이언트로 재 전송해야 하는 단점을 가지고 있으며 전송된 데이터에 대한 멀티미디어 표현 기능을 포함한 모듈도 전송해야 하므로 네트워크 트래픽을 가중 시키는 단점을 지니고 있다. 또한 CGI 방식은 정적으로 구현되기 때문에 멀티미디어 데이터를 포함하는 상품의 정보를 다양한 형태로 보여 주지 못한다

3. 멀티미디어 클래스 설계 기법 #1

본 기법은 멀티미디어 객체를 상품 객체에 포함하여 복합 객체를 만들어 클라이언트의 사용자 어플리케이션이 서비스를 요구 시, 멀티미디어 객체를 포함한 상품 객체만을 클라이언트로 전송하는 방식이다.

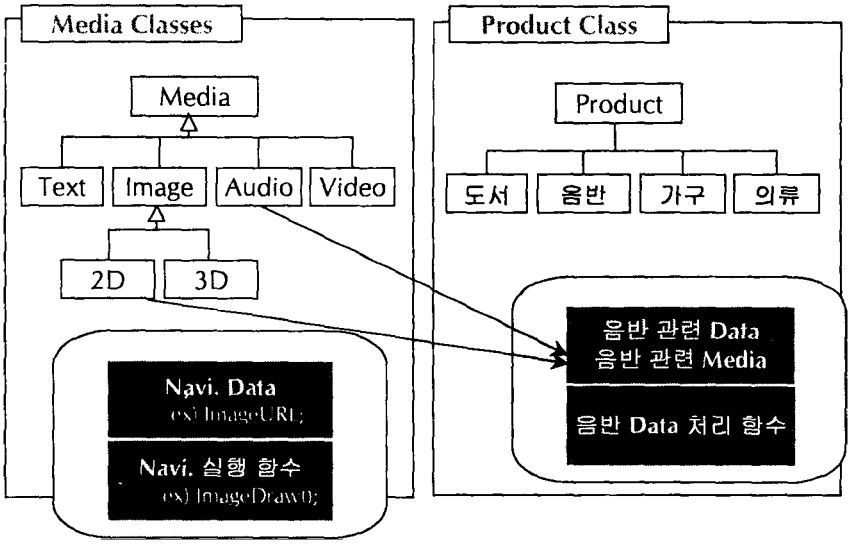

그림 2. Product Class에 Media Class를 통합한 상품 검색

그림 2 에서와 같이 Product 클래스 중 음반 클래스는 오디오와 음반에 관련된 이미지를 보여주기 위해 Media 클래스의 Audio 클래스와 Image 클래스에서 상속받은 2D 클래스를 이용하여 음반 클래스에 포함시켜 복합 객체를 만들어 사용자가 음반에 관련된 정보를 요구 시 Product Class 중 음반 클래스만을 클라이언트 프로그램에서 호출하여 간단하게 멀티미디어 컴포넌트를 설계할 수 있다.

```
Class Audio {
        ....
        Play( );
        VolumeUp( );
        VolumeDown( );
        .... }

Class 2D {
        ....
        ChangeSize( );
        Paint( );
        .... }
```

```
Class 음반 extends Product {
        ....
        Audio audio;
        2D   2d;
        AudioPlay( )
              { audio.Play( ); }
        2DPaint( )
              { 2d.Paint( ); }
        .... }

Class Client extends Applet {
        음반 record;
        .....
        record.AudioPlay( );
        .....
        record.2DPaint( );
        .....
}
```

코드 1. Product Class와 Media Class의 통합 코드

코드 1 에서와 같이 음반 클래스는 Media 클래스의 Audio 클래스와 2D 클래스를 애트리뷰트(Attribute)로 포함시켜 클라이언트 애플릿에서는 음반 클래스만을 객체를 생성시켜 Media 클래스에 있는 기능을 이용할 수 있다. 이때 개발자는 음반 클래스의 인터페이스만을 알고 있으며 내부적인 Media 클래스를 알 필요가 없다.

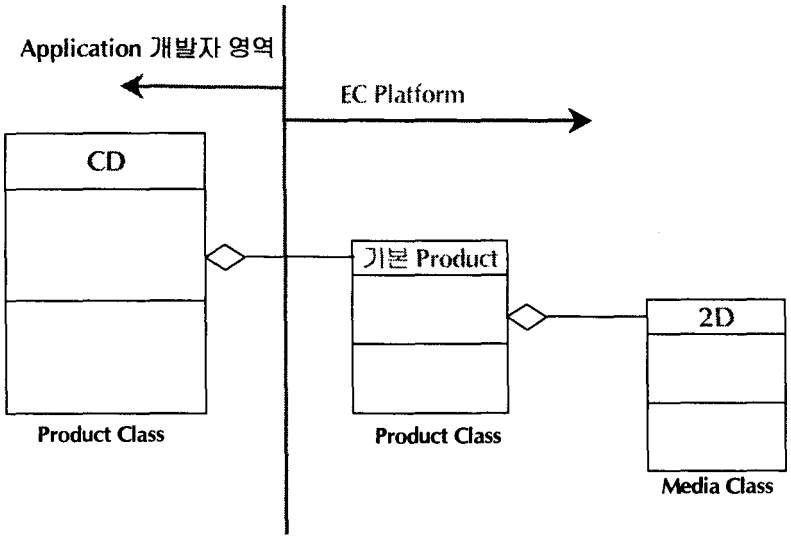

그림 3. 멀티미디어를 위한 EC Platform

그림 3 에서와 같이 어플리케이션 개발자는 기본 Product 클래스만을 알고 있으면 빠르고 쉽게 멀티미디어 웹 어플리케이션을 개발할 수 있다. 이러한 형태의 개

발은 개발자에게 관련 분야의 상품 클래스만을 이용해 개발 절차를 단순화 시킬 수 있다.

서버는 상품 객체의 참조(Reference)를 클라이언트로 전송하는 것이 아니라, 멀티미디어 객체가 포함된 상품 객체 자체를 클라이언트로 전송하므로 요구된 상품 객체의 초기 설정 시간이 많이 소요될 수 있다는 단점이 있지만, 한번 전송된 상품 객체는 클라이언트에 계속적으로 존재하므로 재 전송할 필요가 없으므로 네트워크 트래픽을 많이 줄일 수 있다. 또한, 어플리케이션 개발자들은 멀티미디어 객체를 고려하지 않고, 단지 상품 객체만을 고려하여 개발할 수 있다. 그러나 개발자에게 개발의 다양성을 제공하기 위해 한 상품에 대해 여러 형태의 상품 객체를 구성해야만 한다.

4. 멀티미디어 클래스 설계 기법 #2

본 기법은 멀티미디어 객체를 상품 객체에 포함시키지 않고, 클라이언트 어플리케이션에서 두 객체를 사용하여 상품에 대한 멀티미디어를 구현할 수 있는 방식이다.

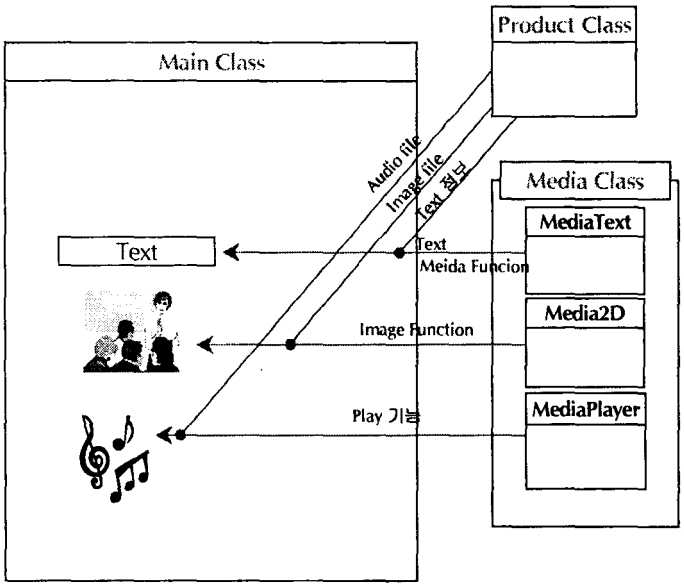

그림 4. Product Class와 Media Class의 분리 형태

그림 4 에서와 같이 클라이언트 어플리케이션 프로그램은 Product 클래스와 Media 클래스를 이용하여 설계할 수 있으며, Product 클래스에서는 그 상품과 관련된 정보들을 내포하고 있고 Media 클래스는 Product 클래스의 정보를 표현할 수 있

는 방법을 제공한다. 이러한 형태의 설계 기법은 개발자에게 다소 번거로운 작업을 요구하지만 개발자에게 다양한 형태의 상품 객체를 구현할 수 있는 플랫폼을 제공한다.

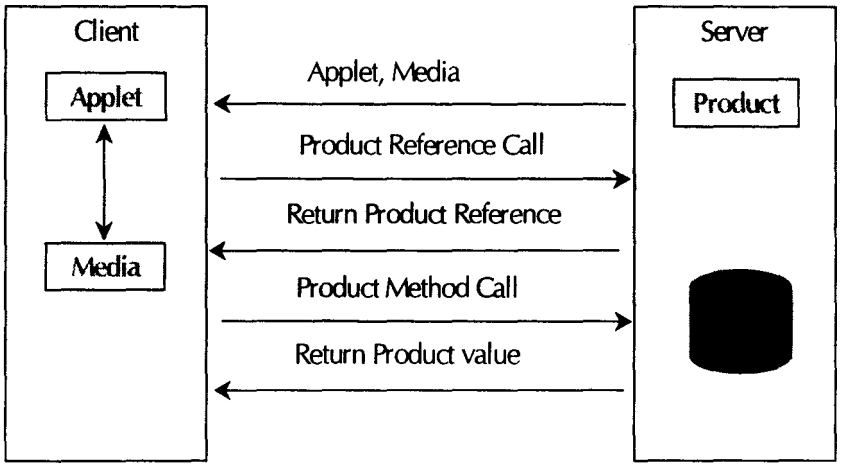

그림 5. 분리된 Product Class와 Media Class을 이용한 상호 작용

그림 5 에서와 같이 클라이언트에서 서버로 상품에 대한 멀티미디어 서비스 호출을 요구 시 서버는 애플릿과 Media 클래스만을 전송하고 클라이언트 호스트에서 Media 클래스의 오퍼레이션을 이용할 수 있으며, 이때 이용되는 데이터는 Product 클래스의 데이터를 호출하여 실행된다. 클라이언트는 원하는 Product 클래스의 참조자(Reference)를 호출하고 서버는 호출한 Product 클래스 참조자의 데이터 제공 함수들을 사용하여 서버의 데이터베이스에 있는 데이터를 호출하거나 저장한다. 즉, 상품에 대한 실제적인 내용이나 멀티미디어 정보를 Product 객체의 오퍼레이션을 통해 가져오고, 이렇게 전송해온 정보를 Media 객체의 오퍼레이션을 통해 동적으로 표현할 수 있다. 전의 Product 클래스와 Media 클래스의 통합 기법과 비교하여 이 기법은 서버로부터 상품의 참조자를 호출하여 전송하므로 초기에 빠르게 네트워크 연결이 이루어 질 수 있다. 그러나 빈번하게 멀티미디어 데이터를 요구하는 경우에는 매번 Product 객체의 데이터를 가져와야 하므로 네트워트 트래픽이 커질 수 있는 단점이 있다. 이렇게 네트워크 트래픽이 빈번한 경우에는 전의 기법에서 처럼 Media 클래스를 포함한 Product 클래스를 모두 전송하는 방식을 적용하여 네트워크로 인한 성능의 저하를 막을 수 있다. 또한 개발 시 개발자는 상품 객체와 멀티미디어 객체를 모두 고려해야 하는 번거로움이 있을 수 있다.

5. 실험 및 평가

위에서의 두 가지 기법 중 Media 클래스와 Product 클래스를 분리하여 클라이언트에서 유연성 있게 Product를 구현하는 사례 연구를 통해 그 기법의 장단점을 알아 보고 웹 어플리케이션을 개발 시 어느 정도 효율적인지를 알아 본다.

```
Class MediaText {
        .....
        public MediaText(String p, int width, int height) { . . . }
        public void ChangeFont(String s, int sz) { . . . }
        public void ChangeColor(Color c) { . . . }
        public void paint( Graphics g ) { . . . }
}
Class Media2D  {
        public Media2D() { . . . }
        public Media2D( String img, mediaplayApplet mapplet) { . . . }
        public Media2D(  int width, int height) { . . .}
        public void setImage( ) { . . . }
        public void paint( String img) { . . .}
}
Class MediaPlayer {
        public MediaPlayer(URL mediaURL, mediaplayApplet mapplet,
                boolean mute, Panel mediaPanel,int mappletWidth, int mappletHeight) { . . . }
        public synchronized void start() { . . . }
        public synchronized void stop() { . . . }
        public synchronized void setVolumeLoud() { . . . }
        public synchronized void setVolumeSoft() { . . . }
        public synchronized void setMute(boolean bool) { . . . }
        public synchronized void controllerUpdate(ControllerEvent event) { . . . }
}
```

코드 2. Media Classes

```
Class Product {
        . . . .
        public void setProduct_Id(String id) { . . . }
        public void setProduct_Image(String image_URL) { . . . }
        public String getProduct_Id() { . . . }
        public String getProduct_Image() { . . . }
}
```

코드 3. Product Class

코드 2는 Media 클래스의 텍스트 정보를 표현하는 MediaText 클래스와 이미지 정보를 표현하는 Media2D 클래스, 그리고 오디오 정보를 실행할 수 있는 MediaPlayer 클래스로 구성되어 있으며 코드 3의 Product 클래스에서 호출한 상품에 관한 정보를 Media 클래스들을 이용하여 실행한다.

```
Class mediaplayApplet extends Applet  {
        MediaText mt;
        Media2D md;
        MediaPlayer mplayer;
        Product p;

public void init() {
        p = new Product();
        String pid = p.getProduct_Id();
        String pname = p.getProduct_Name();

        String s = p.getProduct_Text();
        mt = new MediaText(s,10,100);
        mt.ChangeFont("Courier",17);

        String img = p.getProduct_Image();
        md = new Media2D(img,this);
        md.paint(img);

        String mediaFile = p.getProduct_Audio();
        mplayer = new MediaPlayer(mediaFile, this, mute,
        mainPanel, mappletWidth, mappletHeight);

}
        public synchronized void start() { mplayer.start(); }
        public synchronized void stop() { mplayer.stop(); }
}
```

코드 4. Client Applet Class

코드 4는 클라이언트에서 상품 정보를 보일 수 있도록 구현된 애플릿 코드로 독립적으로 Media 클래스와 Product 클래스를 애트리뷰트로 첨가하여 Product 클래스에서 관련 상품의 정보를 얻는다. 예를 들면, pid = p.getProduct_Id( ), img = p.getProduct_Image( ), 이렇게 얻어진 정보는 Media 클래스(MediaText, Media2D, MediaPlayer)의 오퍼레이션을 이용하여 표현할 수 있다. 예를 들면, md = new Media2D(img, this)... md.paint(img), mplayer = new MediaPlayer(mediaFile)... mplayer.start().

위의 연구에서 미리 설계된 멀티미디어 컴포넌트를 이용해 웹 어플리케이션을 구현해 보았으며 그림 6 에서와 같은 화면의 결과를 나타내고 있다.

Media 클래스와 Product 클래스를 분리한 형태로 웹 어플리케이션의 개발은 다양한 형태의 멀티미디어 상품을 구성할 수 있다. 예를 들면, 음반 상품은 Product 클래스에 MediaText 클래스만을 결합하여 아주 가볍게 구성하여 네트워크 트래픽을 최소로하여 빠른 정보의 제공을 목적으로 할 수 있다. 반대로, Product 클래스에 음반의 상품 광고 비디오를 보여 주고, 음반의 표지 화면을 이미지로 보여 주기 위해 MediaPlayer 클래스와 Media2D 클래스를 결합하여 멀티미디어 정보를 제공할 수 있다. 그러나 코드에서 보는 바와 같이 개발자는 플랫폼에 존재하는 멀티미디어 컴포넌트들의 종류와 각각의 컴포넌트들의 매개변수와 결과값 반환 형태, 그리고 어

떤 멀티미디어 형태들을 제공하는지 자세히 알고 있어야 한다. 또한 그림 6에서 입력한 상품 ID를 이용해 원하는 상품을 검색할 수 있다. 상품 ID를 입력했을 때, 관련 ID의 텍스트 정보, Image URL, 그리고, Audio URL 등은 Product 클래스를 통해서 가지고 오지만, 표현은 Media 클래스를 이용하므로 서로의 클래스들 사이의 관계로 인한 복잡성을 야기할 수 있다.

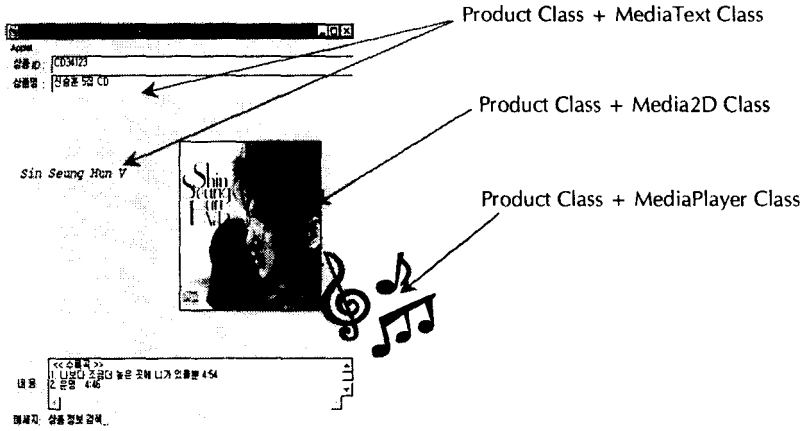

그림 6. 결과 화면

6. 결 론

이상의 두 가지 기법을 이용하여 전자 상거래 플랫폼 개발을 위한 멀티미디어 컴포넌트 설계기법을 살펴 보았고, 각각의 장단점을 고려하여 개발하려고 하는 어플리케이션의 상황에 따라 적절하게 적용하는 것이 두 기법의 효율성을 향상시킬 수 있으며, 결과적으로 어플리케이션의 네트워크 성능을 향상시킬 수 있다.

향후 연구 과제는 어플리케이션의 특성에 따라 Product 클래스에 Media 클래스를 결합하여 복합객체를 만들어 Product 클래스만을 이용하거나 Product 클래스와 Media 클래스를 분리하여 제공할 수 있는 선택적인 개발 설계 기법을 연구한다.

참고 문헌

[1] Mowbray, T. and Malveau, R., *CORBA Design Patterns*, John Wiley & Sons, Inc., 1997.

[2] Orfali, R., Harkey, D. and Edwards, J. *The Essential Client/Server: Survival*

Guide, 2nd Edition, John Wiley & Sons, Inc., 1996.

[3] Pressman, R., *Software Engineering: A Practitioner's Approach*, McGraw-Hill International Press, 1997.

[4] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W., *Object-Oriented Modeling and Design*, Prentice-Hall International, 1991.

[5] Schach, R. *Software Engineering with JAVA*, IRWIN, Inc., 1997.

저자 소개

김 수 동

1984, 전산학 학사, Northeast Missouri State University
1988, 전산학 석사, The University of Iowa
1991, 전산학 박사, The University of Iowa
1991 - 1993 한국통신 연구개발단 연구실장/선임연구원
1993 - 1994 The University of Iowa, 교환교수
1994 – 1995 현대전자 S/W 연구소 책임연구원
1995.9 - 현재 숭실대학교 컴퓨터학부 조교수
주요관심분야
객체지향 개발방법론 (UML, Java)
웹 기반 분산 객체 컴퓨팅 (CORBA, Intranet, Client-Server Web, Web Agents)
인터넷 상거래 시스템 기술

김 철 진

1996. 전산학 학사, 경기 대학교
현재. 숭실대학교 컴퓨터학부 대학원 (석사과정)
주요관심분야
객체지향 사용자 인터페이스
객체지향 개발방법론 (UML, Java)
웹 기반 분산 객체 컴퓨팅 (CORBA, Client-Server Web)