자바기반 분산시스템을 위한 통합개발환경의 구현

°이지현, 유철중, 장옥배
전북대학교 컴퓨터과학과

Implementation of Integrated Development Environment for Java-based Distributed System

°Ji-Hyun Lee, Cheol-Jung Yoo, Ok-Bae Chang
Dept. of Computer Science, Chonbuk National University

요 약

분산 객체 애플리케이션은 모듈화된 설계로 인해 컴포넌트 개발 및 수정을 독립적으로 이루어지도록 해준다. 즉, 분산 객체 시스템은 독립적으로 작동될 수 있으나 다른 컴포넌트와 상호작용이 가능한 모듈로 분리되어 있다. 이러한 분산 객체 시스템에서 객체 편집이나 프로젝트 관리, 컴파일, 레지스트리 구동 등을 위한 통합 개발 환경은 사용자가 보다 편리한 환경에서 작업할 수 있도록 도움을 준다. RMI는 CORBA, DCOM에 비하여 사용자 그룹이나 각종 개발 환경들이 풍부하지 못하지만 자바에 익숙한 프로그래머라면 손쉽게 분산 객체 시스템의 동작을 재사용 컴포넌트들을 시험해 볼 수 있다는 장점이 있다. 본 논문은 이러한 RMI를 이용한 분산 객체 시스템에서 사용자가 각종 인터페이스나 클라이언트/서버 애플리케이션을 작성하고, 작성된 애플리케이션 컴파일 및 디버깅을 수행한 후 작성한 클라이언트/서버 애플리케이션이 의도한 바와 같이 동작하는지 검증 및 확인을 위해 구현된 통합 개발 환경인 JDAT(Java Distributed Application Tester)의 구성과 세부적인 기능, 그리고 용도를 설명하고자 한다

1. 서 론

분산 객체는 원격지의 클라이언트가 메소드 호출을 이용하여 엑세스될 수 있는 독립적인 코드의 조각으로 이루어진 패키지이다. 분산 서버 객체를 만들기 위해 사용되는 언어와 컴파일러는 클라이언트에게 투명하다. 즉 클라이언트는 분산 객체의 위치나 운영체제 환경을 알 필요가 없다. 다시 말해서 분산 객체는 특정 프로그램, 컴퓨터 언어, 또는 구현 환경에 제한을 받지 않는 독립적인 소프트웨어 컴포넌트이다[1]. 이러한 분산 환경을 지원하기 위한 소프트웨어 구조로는 OMG가 제정한 이기종 분산환경에서 애플리케이션 상호연동을 위한 OMA의 핵심기술인 CORBA와 MS 제품군 대부분에 적용된 DCOM, 그리고 JDK에 포함되어 있어 자바 가상 기계가 설치된 시스템에서 동작이 가능한 RMI가 있다. 이미 CORBA, DCOM은 다양한 분야에서 응용되고 있으며 개발 도구들 또한 여러 종류가 개발되어 있다.

본 논문에서는 이러한 분산환경에서 애플리케이션 개발을 위한 소프트웨어 구조들 중 자바 RMI를 이용한 분산 객체 설계 시 객체 편집 및 컴파일, 디버깅, 그리고 객체 서비스 점검 및 확인을 목적으로 구현한 통합 개발 환경인 JDAT의 구성과 각 모듈들의 기능에 대해 논의하고자 한다[2][3]

먼저 2장에서는 자바 분산 객체 애플리케이션 개발에 이용될 수 있는 각종 개발 도구와 개발 환경을 소개하고 JDAT의 특징과 이점, 그리고 JDAT의 주요 구성요소와 기능을 설명하고 3장에서는 사용사례와 JDAT를 이용한 분산 객체 애플리케이션 개발 과정을 설명하고 4장에서는 결론 및 향후 연구과제를 제시한다.

2. 분산 객체 애플리케이션을 위한 통합 개발 환경

2.1 JDAT의 필요성

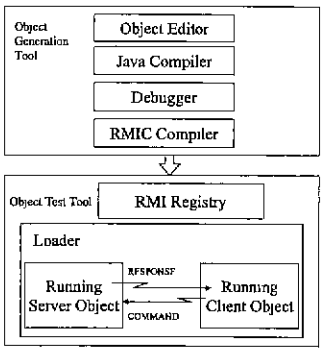
현재 자바를 이용한 응용 프로그램 개발 시 많이 이용하는 개발 도구로는 대표적으로 JDK가 있으며 자바 통합 개발 환경인 KAWA, 그리고 자바 개발 도구인 JBuilder, Bluette, Visual Cafe 등이 있다. JBuilder나 Visual Cafe 등과 같은 통합 개발 도구는 자바 애플리케이션 개발을 위한 각종 컴포넌트와 프로젝트 관리, 컴파일, 디버깅 기능 등의 다양한 갖추고 있다. 이러한 개발 도구의 선택은 개발 목적에 따라 다르겠지만 자바 기반 분산 API인 RMI를 이용하여 작성된 애플리케이션을 실행하기 위한 레지스트리 구동이나 서버 애플리케이션 실행 및 서버 애플리케이션의 동작에 대한 점검 및 객체 유지보수 기능은 갖추고 있지 않다. 따라서 RMI를 이용하여 분산 객체 애플리케

이션을 작성하고자 하는 개발자들은 기존의 통합 개발 도구를 이용하여 객체를 작성한 후 객체의 동작을 검증하기 위해 거쳐야 하는 여러 과정을 별도로 실행해야만 하는 불편함을 감수해야 한다 따라서 이러한 과정을 통합하기 위한 통합 개발 환경의 개발은 필수적이며 개발된 객체의 동작을 검증하기 위한 테스트 및 감시 기능 또한 필요하다

2.2 JDAT의 구성

JDAT는 크게 애플리케이션 편집 및 컴파일, 그리고 디버깅 기능을 포함하는 객체 생성 도구와 레지스트리 구동 및 객체 실행 및 모니터 기능을 포함하는 객체 테스트 도구로 나누어 볼 수 있다. 본 절에서는 JDAT의 구성을 주요 모듈 별로 나누어 설명하고자 한다.

[그림 1]은 JDAT의 주요 구성요소를 보여주고 있다



[그림 1] 통합 개발 환경의 구조

2.2.1 객체 생성 도구

객체 생성 도구는 객체 편집과 디버깅 윈도우를 통한 오류 확인 및 정정, 그리고 프로젝트 윈도우를 통한 프로젝트 관리를 수행할 수 있으며 작성된 객체를 컴파일 할 수 있게 해준다. 본 도구는 JDK의 'javac'와 'rmic' 컴파일러를 이용하므로 100% 순수 자바 애플리케이션을 개발할 수 있다. 각 모듈의 기능을 정리하면 [표 1]과 같다.

[표 1] 객체 생성 도구의 구성요소와 기능

도구	기 능
Project Manager	분산 객체 인터페이스와 구현객체를 통합관리 유지.
Object Editor	분산 객체 인터페이스 정의나 구현, 그리고 클라이언트 클래스를 편집/수정.
Java Compiler	클라이언트/서버 애플리케이션을 바이트 코드로 변환.
Debugger	각종 오류 확인 및 정정
RMIC Compiler	클라이언트 스터브와 서버 스켈리턴을 생성.

2.2.2 객체 테스트 도구

객체 테스트 도구는 객체 생성 도구를 통하여 컴파일 완료된 바이트 코드를 테스트하기 위해 레지스트리 구동 및 작성된 서버 측 애플리케이션을 실행한다 서버 애플리케이션의 동작을 점검하기 위해 작성된 클라이언트 애플리케이션을 실행한 후 서버 애플리케이션의 서비스 상태를 출력 윈도우를 통해 확인하며 테스트할 수 있다

각 모듈의 기능을 정리하면 [표 2]와 같다.

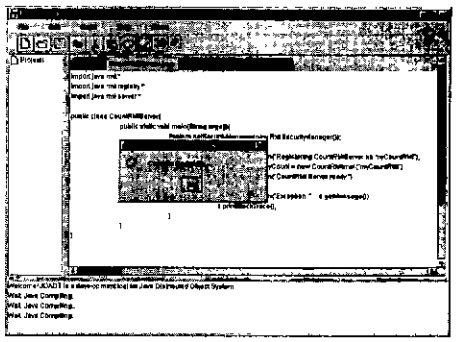
[표 2] 객체 테스트 도구의 구성요소와 기능

도구	기 능
RMI Registry	RMI 레지스트리를 구동.
Loader	서버 애플리케이션 구동 및 클라이언트 애플리케이션 실행을 통한 서비스 상태 점검 및 확인

3. JDAT의 사용자 인터페이스

본 절에서는 구현된 JDAT의 사용자 인터페이스와 실제 사용사례를 들어 설명하고자 한다. 초기 인터페이스는 프로젝트 윈도우, 편집 윈도우, 그리고 디버깅 윈도우로 구성되어 있다. 사용자는 편집 윈도우를 통해 인터페이스나 객체를 작성하고 프로젝트 윈도우를 통해 애플리케이션을 통합 관리할 수 있다. 또한 컴파일 결과나 실행 상태를 디버깅 윈도우를 통해 확인할 수 있다.

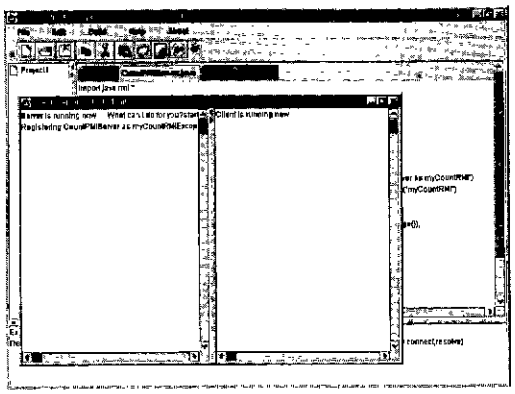
[그림 2]는 JDAT의 프로젝트 윈도우, 편집 윈도우, 디버깅 윈도우를 보여주는 인터페이스 환경이며, 작성된 객체를 컴파일하고 컴파일이 완료되었음을 보여주고 있다.



[그림 2] JDAT를 이용한 객체 편집 및 컴파일

이렇게 컴파일된 서버 애플리케이션은 클라이언트의 메소드 호출에 따라 정확한 서비스를 수행할 수 있는지 여부를 테스트해야 한다.

[그림 3]은 객체 생성 도구를 통하여 생성된 서버 애플리케어션의 동작을 테스트하는 과정에서 클라이언트와 서버의 서비스 요구와 응답 상태를 보여주고 있다



[그림 3] JDAT를 이용한 클라이언트/서버 애플리케이션 테스트

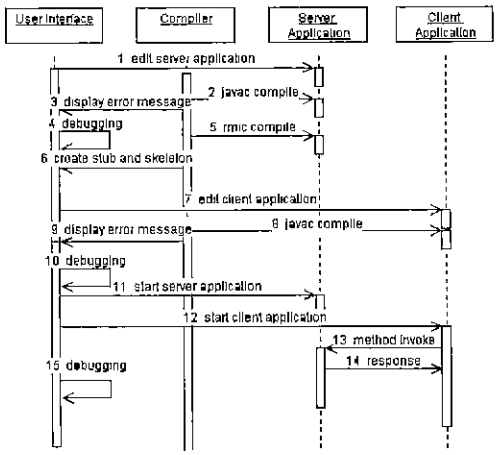
이러한 과정을 통해 개발자는 보다 편리한 환경에서 분산 객체 애플리케이션을 작성하고 테스트할 수 있다.

3.2 JDAT의 사용사례

JDAT를 이용하여 분산 애플리케이션을 작성하고 컴파일한 후 개발한 애플리케이션의 디버깅과 동작 테스트 과정을 다음과 같이 정리하여 볼 수 있다.

- JDAT 편집기를 이용한 서버 애플리케이션의 인터페이스와 인터페이스 구현 코드 작성.
- 서버 애플리케이션을 테스트할 클라이언트 애플리케이션 작성.
- 서버 애플리케이션의 인터페이스와 인터페이스 구현 코드를 자바 컴파일러를 이용하여 컴파일 및 디버깅.
- 서버 애플리케이션의 인터페이스 구현 코드 RMI 컴파일(클라이언트 스터브와 서버 스켈리턴 생성).
- 클라이언트 애플리케이션 작성, 컴파일 및 디버깅(객체가 분산되어 있는 경우에는 서버 애플리케이션 구현 코드를 rmic 컴파일한 결과 생성된 스터브 클래스를 서버로부터 복사).
- 서버 애플리케이션의 이름을 등록시키기 위한 레지스트리 구동(서버 측에서만 구동).
- 서버 애플리케이션 구동 및 테스트를 위한 클라이언트 애플리케이션 실행.
- 서버 애플리케이션 응답 점검 및 디버깅[4]

[그림 4]는 JDAT를 이용하여 분산 애플리케이션 작성 및 테스트 과정을 보여주는 순차 다이어그램이다



[그림 4] JDAT 사용사례 순차 다이어그램

4. 결론 및 향후 연구과제

클라이언트/서버 애플리케이션을 개발하는데 있어서 객체 편집이나 컴파일 뿐만 아니라 객체의 동작 확인 및 검증, 그리고 지속적으로 동작 상태를 모니터하기 위한 개발 도구는 필수적인 것이다 본 논문에서는 이러한 기능을 지원하기 위한 통합 개발 환경인 JDAT를 구현하였다

JDAT는 단순히 편집이나 디버깅, 프로젝트 관리 뿐만 아니라 서버 측 객체의 동작 상태를 모니터할 수 있도록 해주며 클라이언트 애플리케이션이 서버에 접속하여 올바른 서비스를 받을 수 있는지 점검할 수 있게 해준다. JDAT는 RMI를 이용하여 분산 시스템을 실습하고자 하는 학생이나 분산 애플리케이션을 작성하는 프로그래머에게 도움을 줄 수 있다.

향후 JDAT에 사용자 편의를 위한 각종 도움말 제공 기능과 JDBC 구동 기능 등을 포함시켜 보다 완벽한 자바 분산 애플리케이션을 개발하기 위한 통합 개발 환경이 되도록 할 예정이다

참고문헌

[1] Robert Orfali, Dan harkey, Jeri Edwards, The Essential Distributed Object Survival Guide, John Wiley & Sons, Inc, 1996.

[2] 이지현, 진형수, 장근실, 유철중, 장옥배, "자바기반 분산시스템을 위한 통합 개발 도구의 설계", 한국정보과학회 '99 봄 학술발표논문집(A), 제 26권 1호, pp. 590~592, 1999

[3] Best practice in distributed object application development: RMI, CORBA and DCOM, http://www.developer.com/news/techfocus/022398_dist1.html, 1998

[4] Java™ Remote Method Invocation Specification, 1998