자바객체를 사용할 수 있는 자바스크립트 해석기의 설계 및 구현

이창환, 오세만
동국대학교 컴퓨터공학과

Design and Implementation of the JavaScript Interpreter to Use Java Objects

Changhwan Yi, Seman Oh
Dept, of Computer Engineering, Dongguk Univ

요 약

사용자와 개발자의 컴퓨터 환경은 시간이 지날수록 객체지향 환경으로 변화하고 있다. 초기에는 소프트웨어 개발단계에 객체지향 기술이 적용이 되었으나, 최근에는 운영체제와 같은 사용환경에도 객체지향 기술이 적용되고 있다 또한 과거 운영체제에서는 순차적이고 반복적인 작업을 쉽게 하기 위해 스크립트 언어를 많이 사용하였으나 이와 같은 스크립트 언어는 객체지향 방법이 적용된 사용환경에서는 적합하지 않은 문제점이 있다 따라서 객체지향 기술을 사용할 수 있는 스크립트 언어에 대한 요구가 증가하였고, 자바스크립트도 그런 요구의 일부를 만족시키는 언어이다. 그러나 현재까지 사용되는 자바스크립트 해석기는 시스템에서 제공되는 객체만이 사용 가능했으며 언어명세에도 기능확장을 위한 방법이 정의되어 있지 않다 따라서 추가적인 기능확장을 위한 방법이 제공되지 않는 단점이 있다.

본 논문에서는 자바객체를 사용할 수 있는 자바스크립트 해석기를 설계하고 구현하였다. 제안한 자바스크립트 해석기는 사용자가 정의한 자바객체를 사용할 수 있으며, 기능 확장이 불가능한 자바스크립트의 단점을 극복할 수 있다.

1. 서론

소프트웨어 위기를 해결하기 위한 방법의 하나로 나타난 객체지향 기술은 시간이 지날수록 적용 범위가 소프트웨어 개발단계뿐만 아니라 컴퓨터의 여러 분야로 넓어지고 있다. 이 분야 중 하나가 운영체제와 같은 사용환경이다. 사용환경에 객체지향 기술이 적용되고 있으나, 이전 사용환경에서 사용했던 스크립트 언어는 순차적이고 반복적인 작업에 알맞을뿐 객체지향 방법이 적용된 곳에서는 사용이 적합하지 않는 문제점이 있다. 그래서 객체지향 방법을 쉽게 사용할 수 있는 스크립트 언어에 대한 요구가 증가하게 되었는데 새로운 스크립트 언어에 요구를 만족시키기 위해 여러 가지 언어가 소개되었다 이 중 웹(WWW)에서 시작한 스크립트 언어인 자바스크립트도 이런 요구를 충족시키고, 사용에 익숙한 많은 개발자가 있어 웹만이 아닌 일반적인 사용환경에서도 자바스크립트의 사용이 증가하고 있다.

그러나 현재 사용되는 자바스크립트 해석기는 시스템에서 제공되는 객체만이 사용 가능하다. 이는 언어명세에도 기능확장에 대한 방법이 정의되어 있지 않기 때문이며, 따라서 추가적인 기능확장을 위한 방법을 제공할 수 없다는 단점이 있다

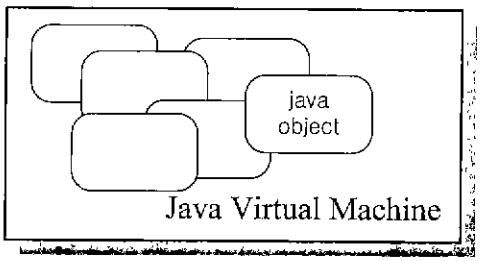
본 논문에서는 사용자가 정의한 자바객체를 사용할 수 있도록 하여, 기능 확장이 불가능한 자바스크립트의 단점을 극복할 수 있는 자바스크립트 해석기를 설계하고 구현하였다.

2. 자바객체와 자바스크립트

2.1. 자바객체

객체는 내부에 자신의 상태를 나타내는 정보와 이 내부에 저장된 정보에 대한 연산·접근을 허용하는 함수로 구성되어 있다. 이러한 객체는 객체가 저장되고 관리되는 주체에 따라 구분된다. 이러한 주체는 언어 자체가 될 수도 있으며, 시스템이나 환경이 될 수도 있다.

이러한 구분에 따라 **자바객체(Java objects)**를 정의하면, 자바에 의해서 관리되는 객체를 자바객체로 부를 수 있게 된다. 그러나 자바는 언어적인 측면과 실행 환경적인 측면을 모두 가지고 있다. 즉 객체의 정의 및 사용은 언어를 통해서 하지만, 실행시에는 JVM에 의해서 관리가 되기 때문이다. 그러므로 작은 의미에서는 JVM에 의해서 관리되는 객체를 자바객체로도 부를 수 있을 것이다. 그래서 본 논문에서는 JVM에 의해서 관리되는 객체만을 자바객체로 정의한다.



[그림 1] JVM내에서의 자바객체

2.2. 자바스크립트

자바스크립트는 원래 명칭은 LiveScript로 Netscape사에 의해서 개발된 객체지향언어이다. 개발 당시에는 웹 환경에서 주로 사용되었으나, 웹 환경이 컴퓨터 사용이 주요 환경이 되면서 컴퓨터의 다른 부분에서의 사용도 늘고 있으며, 마이크로소프트사에서는 자바스크립트와 호환이 되는 JScript를 발표했다.

이들을 실행하기 위해 여러 자바스크립트 해석기가 사용되지만, 자바스크립트에 대한 표준화는 ECMA(European Computer Manufacturers Association)내에 있는 TC39 위원회[6]에서 의해서 이루어지고 있는데, 이 위원회에서는 일반적으로 알려진 자바스크립트라는 이름 대신에 ECMAScript라는 이름을 사용한다. ECMAScript는 현재 2번째 표준 언어명세가 발표된 상태이고, 3번째 표준 언어명세에 대한 작업이 진행중이다.

3. 설계

3.1. 개발환경

본 논문에서는 Netscape Navigator 소스를 공개한 사이트인 Mozilla[7]에서 제공하는 자바스크립트 해석기 소스를 수정하여 본 논문에서 제안하는 해석기를 개발하였다. 이 자바스크립트 해석기 소스의 초기 버전은 Netscape Navigator 소스 안에 있는 자바스크립트 해석기를 단순히 분리한 것이었으나 이후 버전에서는 Netscape Navigator와는 독립적으로 개발되고 있고, 브라우저안에서 뿐만 아니라 스탠드 알론(Stand-Alone)으로 사용될 수 있도록 개발되고 있다

- JavaScript Reference Implementation 1.4.2

해석기를 개발하기 위해 사용된 환경은 다음과 같다.

- Java 2 SDK, Standard Edition 1.2.2 (JDK 1.2.2)
- Microsoft Windows 98 Second Edition
- Microsoft Visual C++ 6.0 with Service Pack 3

개발환경은 위와 같지만, JDK 1.1에서 제공하는 JNI(Java Native Interface)와 JDK 1.2에서 제공하는 JNI는 호환이 되기 때문에, 본 논문에서 사용하는 모든 JNI 함수들이 JDK 1.1에서 제공하는 JNI에도 존재한다. 따라서 JDK는 1.1 이상버전만 사용한다면 사용에는 문제점이 없다.

3.2. 제공 기능

본 논문에서 제안하는 자바스크립트 해석기는 이전 자바스크립트 해석기에 다음과 같은 기능을 추가로 제공한다.

- 자바스크립트 프로그램에서 자바객체 생성
- 자바스크립트 프로그램에서 자바객체 사용

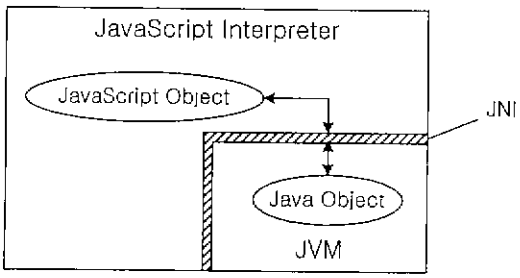
3.3. 작동 구조

JVM에 의해서 관리되는 자바객체를 자바스크립트 해석기에서 사용하기 위해서는 자바객체에 대응하는 자바스크립트객체를 만들고, 이 객체에 대한 사용을 자바객체에 대한 사용으로 변환하는 작업이 필요하다.

자바객체에 대응하는 자바스크립트객체를 만들기 위해서는 자바객체에 대한 정보를 JVM으로 가져올 수 있어야 하고, 이를 위해 자바에서 제공하는 Core Reflection API를 이용한다.

자바스크립트객체에 대한 사용을 실제 자바객체에 대한 연산으로 변환할 때, Core Reflection API를 사용하여 얻은 정보를 이용하여 형을 일치시키는 작업을 한다

그리고 자바객체에 대한 작업을 하기 위해서 자바에서 제공하는 JNI를 이용한다. C/C++ 코드에서 자바객체를 사용하기 위해서는 자바에서 제공하는 JNI를 이용해야 하기 때문이다.



[그림 2] 본 논문에서 제안한 자바스크립트해석기의 작동 구조

4.구현

자바스크립트 해석기에서 자바객체를 사용하기 위해서는 다음과 같은 문제점이 있다. 첫째, 자바스크립트에서 제공하는 자료형(Data type)과 자바에서 제공하는 자료형을 어떤 방법으로 일치시킬 것인지를 결정해야 한다. 둘째, 다른 환경에 있는 객체를 어떤 식으로 연결시킬 것인지를 결정해

야 한다. 셋째, 자바객체 멤버에 대한 연산을 어떻게 자바객체에 대한 연산으로 변환할 지를 결정해야 한다.

위와 같은 문제점을 본 논문에서는 다음과 같은 방식으로 해결을 하였다. 첫 번째 문제는 자바스크립트 해석기에서 내부적으로 유지하는 자료형 정보를 이용하여 자바가 제공하는 자료형으로 변환하는 것이다.

서로간에 변환 가능한 대응 관계는 다음과 같다.

자바스크립트 형	자바 형
Boolean	boolean
String	java.lang.String
Number	int 또는 long
Number	float 또는 double
Object	Object reference

표에서 문제가 되는 것은 자바스크립트의 Number형이 자바의 정수와 실수계열 형에 전부 대응되는 점이다. 그러므로, 어떤 것으로 대응될 지는 자바스크립트 해석기가 가지고 있는 내부 정보를 이용하여 해결하였다 두 번째 문제는 자바스크립트객체 내부에 JNI에서 제공하는 자바객체에 대한 참조(Reference)를 유지하여 해결하였다. 세 번째 문제는 자바객체에 대응하는 자바스크립트객체를 생성할 때, 자바객체 멤버에 대한 정보를 얻어 자바객체 멤버에 1:1로 대응하는 자바스크립트객체 멤버를 생성하는 것이다. 생성된 자바스크립트객체를 통해 자바스크립트객체 맴버를 사용하면, 자바스크립트객체 멤버와 대응하는 자바객체 멤버를 자바스크립트 해석기가 JNI를 통해 사용하도록 했다.

4.1. 자바객체 생성

자바스크립트에서 자바객체를 생성하기 위해서 자바스크립트 해석기는 JavaObject라는 자바스크립트 객체를 제공한다. 이 객체를 통해 새로운 자바객체를 생성하고자 할 때, 자바객체의 클래스 이름을 JavaObject에 전달하여 준다. 그러면 자바스크립트 해석기는 그 이름을 갖는 클래스파일이 존재하는지를 확인한다. 없으면 오류 메시지를 출력한다. 그 이름을 갖는 클래스 파일이 있으면, 자바객체를 생성하고 생성된 객체로부터 멤버에 대한 정보를 추출한다. 추출되는 정보는 필드이면 이름하고 형이고, 메소드이면 이름, 반환형, 인자 수, 인자들의 형이다. 자바객체 외부에서 사용할 수 있는 멤버를 이 정보를 사용하여 자바스크립트에 Property와 Function으로 추가한다. Property와 Function이 추가된 자바스크립트객체를 반환한다. 본 논문에서 제안한 자바스크립트 해석기는 위의 방법으로 자바객체를 생성한다.

4.2. 자바객체 멤버 사용

자바스크립트 프로그램에서 자바객체의 멤버를 사용하기 위해서는 자바객체에 대응되는 자바스크립트 객체의 맴버를 사용하면 된다. 자바객체에 대응되는 자바스크립트 객체의 맴버가 사용되면 자바스크립트 해석기는 자바스크립트 객체의 멤버가 가지고 있는 자바객체에 대한 정보를 사용하여, JNI를 통해 자바객체 멤버를 사용할 수 있었다. 이때, 멤버에 대한 정보를 사용하여 자바스크립트와 자바간에 형을 변환한다. 자바객체 맴버를 사용한 결과와 사용 중에 발생한 예외나 오류를 자바스크립트에 반환한다

4.3. 사용 예

다음은 행렬을 자바 객체를 구현하고, 구현된 자바객체를 사용하는 자바스크립트 프로그램을 본 논문에서 제안한 자바스크립트 해석기에서 사용하는 예를 보인 것이다.

```
m1 = JavaObject("Matrix2By2");
m2 = JavaObject("Matrix2By2");
m1.setValue( 1, 2, 3, 4 );
m2.setValue( 6, 7, 8, 9 );
m1.add( m2 );
str = m1.toString();
print( str );
print( "\n" );
```

이와 같은 자바스크립트 프로그램의 결과는 다음과 같다.

```
[[7, 9] [11, 13]]
```

5. 결론 및 향후연구

자바스크립트 언어명세뿐만 아니라 자바스크립트 해석기에서도 기능확장을 위한 방법을 제공하지 않는 자바스크립트 해석기를 수정하여, 본 논문에서는 자바를 이용하여 기능을 확장할 수 있는 자바스크립트 해석기를 소개하였다.

향후 연구 과제로는 다음 자바스크립트 해석기 명세에 제안된 import와 package 예약어에 대한 정의를 사용하여 자바객체와 자바스크립트에서 제공되는 객체를 같은 방식으로 생성할 수 있는 방법을 연구하려고 한다. 그리고 CORBA나 COM과 같은 다른 객체 시스템의 객체를 자바스크립트 해석기에서 사용할 수 있도록 하는 것을 연구하고자 한다.

6. 참고문헌

[1] The Java Language: An Overview, Sun Microsystems
[2] Java Native Interface, 1997, Javasoft
[3] Java Core Reflection API, 1997, Javasoft
[4] *Campione, Walrath, Huml, and Tutorial team, Java* Tutorial Continued, 1998, Addison-Wesley
[5] ECMAScript Language Specification, 1998, ECMA
[6] http://www2.hursley.ibm.com/tc39, ECMA TC39
[7] http://www.mozilla.org/js, JavaScript
[8] 이창환, 오세만, "C/C++에서 자바객체 이용을 위한 인터페이스", 한국정보처리학회 '99춘계 학술논문발표집, 1999