비동기통신 방식을 이용한 GWT(Google Web Toolkit) 의 데이터 모델 구현

최경영*, 이보름*, 이석희*, 조상*
*청주대학교 컴퓨터정보공학과
e-mail:c1b2a3@nate.com

Asynchronous communication by using GWT assessment and analysis of the data model

Kyoung-Young Choi*, Bo-reum Lee*, Seok-Hee Lee*,
Sang Cho*
*Dept of Computer Information Engineering, Chong-Ju University

요 약

Google에서는 GWT라는 웹2.0시대의 핵심기술인 AJAX를 기반으로 한 개발 툴킷을 발표했다. GWT는 Desktop Application 수준의 웹 시스템을 Java로 구현하여 AJAX Application을 재사용 가능하도록 한다. 이에 따른 비동기 통신 방식에서의 모델(M)과 콘트롤러(C)가 RPC 서비스에 적용되는 방법이 기존의 Java 웹 프레임워크에서 적용한 페이지 단위의 방식인 동기식 통신이 적용되는지 분석하고 비동기 통신 방식에 간단한 디자인패턴을 사용하여 구현 하였을 때 효율성이 기존의 프레임워크보다 높은가를 평가한다.

1. 서론

인터넷의 사용은 웹의 사용이라고 할 수 있을 만큼 인터넷에서 웹의 비중이 크다. 소프트웨어 발전방향을 이야기 할 때 웹을 빼놓고는 이야기 할 수 없을 정도로 웹은 사용분야가 넓고 방대하다. 많은 개발자들의 노력으로 Desktop 어플리케이션 수준의 프로그램을 웹에서도 개발할 수 있게 되었다.

웹에서의 어플리케이션 구현은 ActiveX컨트롤(일명 OCX컨트롤 혹은 OLE컨트롤)이라는 마이크로소프트의 기술로 가능하기도 하지만 Windows라는 운영체제에 종속적이지 않으면서 호환성이 강한 웹 어플리케이션이 필요로 하게 되어 웹2.0시대에 이르게 되었다. 웹2.0의 핵심기술인 Asynchronous JavaScript + XML(Ajax)은 사용자와 웹 페이지 인터랙션을 웹 브라우저와 서버 통신으로부터 분리시켰다.[2] 특히, Ajax는 여러 소스들로부터 온 콘텐트와 서비스들을 결합하여 하나의 통합된 사용자 경험으로 만드는 웹 어플리케이션인 매시업[1]을 실행하는데, 이는 여러 콘텐츠나 서비스를 하나의 사용자 경험(user experience)으로 통합시킨다.[1] 이처럼 사용자 경험을 통합시키고 보다 효율적인 HCI(Human Computer Interactive)를 만들어 내기 위해서는 친숙하면서 단순한 사용자 인터페이스와 알고리즘에 대한 연구가 필요하다.

따라서, 웹 개발에서는 코드의 재활용성과 개발기간의 단축 그리고 안정적인 기술개발이 요구되고 있다. Google web-toolkit(GWT)은 AJAX 어플리케이션을 자바 언어로 작성할 수 있도록 자바 기반 개발 환경을 제공한다. GWT는 XMLHttpRequest [2]객체 API를 캡슐화하여 크로스 브라우저 관련 이슈를 최소화 했음으로 GWT를 사용하면 각 브라우저의 특성에 맞추는 작업을 최소화 할 수 있다.[1] 이 연구에서는 GWT RPC 프레임워크를 사용하여 Web 어플리케이션에 비동기적으로 접근할 때 사용되는 데이터모델을 처리하는 부분을 어떻게 효율적으로 하는가를 분석하고 기존의 Java프레임워크에서 사용하던 통신방

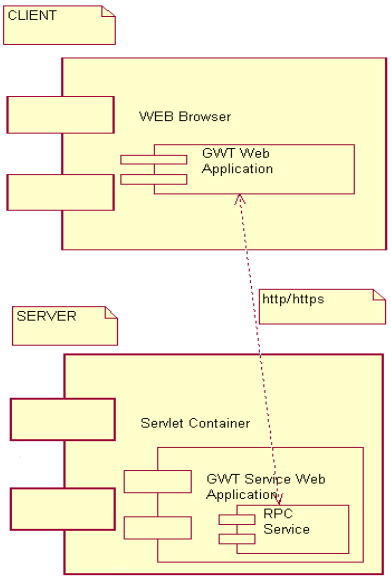
1) 전형적인 매시업 애플리케이션은 하나의 클라이언트 측 애플리케이션으로 되므로, 매시업의 다른 부분들은 정보를 공유하고 DOM 트리나 브라우저 윈도우 프로퍼티 같은 여러 브라우저 리소스를 통해 인터랙팅 한다. 매시업의 특정 부분이 악의적인 의도로 작성될 때(또는 해킹될 때) 악성 코드를 애플리케이션에 투입할 수 있다. 이는 민감한 사용자 정보를 훔치는 것을 포함하여 모든 종류의 공격이 될 수 있다. (XSS와 비슷) [2]

2) XMLHttpRequest는 클라이언트 측 JavaScript가 HTTP를 원격 서버로 연결하여 플레인 텍스트, XML, JavaScript Serialized Object Notation (JSON) 같은 데이터를 교환할 수 있도록 해주는 API이다.

식을 간단한 디자인패턴으로 구현한다.

2. 본론

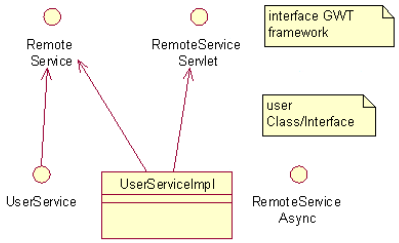
프로그래밍은 최대한 결속력이 없는 객체지향 프로그래밍으로 보수가 쉽고 업데이트가 간편하게 작성하는 노력이 절실하다. 비동기식 통신의 구현은 개발자측면에서는 어떻게 캡슐화 하느냐가 관건이며 또한 데이터를 가져오거나 보낼때 얼마만큼 단순하게 가능한지에 대한 연구가 필요하다. GWT에서는 RPC서비스를 생성하고 구현된 기능을 호출하면 요청한 데이터나 정보를 비동기적으로 받아 페이지 전체를 갱신하지 않으면서 화면에 출력한다. <그림1>은 GWT 어플리케이션의 비동기식 통신 기본 아키텍처다.



(그림 1) GWT의 기본 아키텍처 : SERVER의 RPC SERVICE 와 CLIENT의 비동기 통신

GWT 어플리케이션에서의 서비스는 서버측 모델에 대한 Interface를 제공하며, 대부분의 경우 MVC아키텍처[3]의 모델 부분에 해당한다.

RPC Service는 <그림2>와 같이 GWT 어플리케이션에서의 제공하는 Interface와 사용자의 Class 및 Interface로 구성된다. GWT에서는 Servlet 기반의 접근 방식을 사용하는 경향이 있다. 왜냐하면, 유저서비스는 HttpServlet을 상속한 RemoteServiceServlet을 상속하기 때문이다. RemoteServiceServlet은 입력된 요청의 직렬화된 상태를 복원하고(deserialize) 서버의 처리 결과에 대한 응답을 직렬화하는 작업을 담당한다.



(그림 2) GWT Apllication에서 제공하는 Interface와 사용자 정의 Interface 및 Class

이 부분은 대부분의 MVC아키텍처의 모델(M) 부분에 해당하며 본 논문에서 제시하고자 하는 부분이기도 하다.

<표1> Button Event for Async Result

```
new Button ( new ClickListener() )
 public void onClick(Widget sender)
   AsyncCallback callback = new AsyncCallback()
    public void onSuccess(Object obj)
          .
    public void onFailure(Throwable caught)

  UserService.UserMethod(arg, callback);
          .
```

<표1>과 같이 버튼에 대한 이벤트 핸들러를 구현하고 버튼에 이벤트가 발생 했을 때 UserService를 호출 하게 작성한다.
UserService는 UserMethod의 Prototype을 제공하고 RemoteServiceAsync로 UserServiceImpl과 동기화 된다.
<표2>와 같은 UserServiceImpl에 <표3>와 같이 구현한 Singleton 디자인 패턴을 적용하여 Database혹은 Filebase에 접근하고 데이터를 리턴하는 메서드를 구현한다.

<표2> UserServiceImpl

```
public List getData(String query)
        List data = Singleton.getInstance().query(query);
```

<표1>의 메서드 onSuccess 부분을 <표4>와 같이 객체값을 해당 객체로 캐스팅하여 받는다. 물론 비동기 인스턴스인 callback이 도중에 문제가 발생 한다면 onFailure메서드가 호출 되어진다.

3) MVC(Model-View-Controller)아키텍처 패턴 (1) Model : 모든 데이터, 상태 및 어플리케이션 로직이 들어있다. (2) View : 모델을 표현하는 방법을 제시한다. (3) Controller : 사용자에게 입력받아서 그것이 모델에게 어떤 의미가 있는지 파악한다.

<표3> Singleton Class

```
Class Singleton
  Singleton instance = null;
  private Singleton()
          //empty
  public static Singleton getInstance()
          if instance = null then
          instance = new Singleton();
          end if
          return instance;
  public List getDb(String query)
          Connection c
          Statement st
          ResultSet rs
          rs = c.st(query)
          List list
          while(rs.hasNext())
            list.add(rs.next())
          return list
  public List getFb(String filename)
          String readline
          File f = new File(filename)
          List list
          while( (readline = f.readLine()) != null )
          return list
```

<표4> get result

```
  public void onSuccess(Object obj)
          List list = (List)obj
```

Service 메소드의 리턴 타입은 List가 아닌 다른 객체를 사용해도 된다. Database혹은 Filebase에 접근하는 부분이 요청을 받는 부분은 서비스 시작부터 종료까지 단일 인스턴스인 싱글톤으로 구현되었으며 UserService에서는 단 한 줄을 사용하여 결과 값을 얻을 수 있게 된다.

3. 결론

웹 서비스는 고속통신망의 증가로 점점 자원이 풍부해지고 있다. 본 연구는 GWT RPC서비스의 비동기 통신 방식을 분석하고 코드를 결속력이 낮게 단순화하기 위한 연구이다. 따라서, 속도보다는 실제 개발자가 구현하는데 있어서의 재사용성에 비중을 두었다.

RPC서비스에서 외부 클래스인 Singleton을 임포팅하고 Database나 Filebase에 접근하고 결과를 리턴하는 방법으로 구현하였다. 본 연구에 사용한 클래스는 새로운 GWT 프로젝트에 쉽게 응용하여 사용 할 수 있다. 또한, 단일 인스턴스로 Connection에 대한 메모리 누적을 예방할 수 있으며 인스턴싱의 시간을 절약 할 수 있다. 그리고 비동기화 속에 동기화를 유지해서 동시간대에 다른 유저가 접속해서 인스턴스를 생성하는 것을 막을 수 있으며 호출이 간편하고 결속력이 적어 메소드를 쉽게 변경하여 사용하는 것이 가능하다. 기존의 동기식 통신방식인 Sturts 프레임워크나 Spring 프레임워크의 MVC 아키텍쳐와 비교 했을 때 모델부분이 아닌 뷰에서 부분 리로딩(Reload) 방식으로 인하여 미세한 속도가 올라간다. 또한 Singleton 디자인 패턴을 구현함으로써 다른 디자인 패턴을 적용하는 것도 쉽다고 예상된다. 그리고 RPC 서버의 비동기식 통신 방식의 보안상 취약 한 점을 보완하기 위한 JSON을 이용한 방법등을 쉽게 첨부 할 수 있다. JSON은 단순한 괄호 형식의 구조를 가진 플레인 텍스트이기 때문에, 많은 채널들이 JSON 메시지를 교환할 수 있다. Same-Origin 정책 때문에, 여러분은 외부 서버와 통신할 때 XMLHttpRequest를 사용할 수 없다. JSON with Padding (JSONP)는 JSON과 <script> 태그를 결합하여 사용함으로써 Same-Origin 정책을 우회하는 하나의 방식이다.[2]

기존의 싱글톤 디자인 패턴이 지니고 있던 단점도 있다. 대표적인 예로 양이 적지만, 클래스의 인스턴스는 개체가 참조를 요청할 때마다 이미 존재하는지를 확인해야하는 오버헤드가 있다. 이러한 단점은 인스턴스의 정적초기화 방법으로 해결방법이 제시되어지고 있다.

참고문헌

[1] Google Web Toolkit: GWT Java Ajax Programming - Prabhakar Chaganti

[2] http://www.ibm.com/developerworks/kr/library/x-ajaxsecurity.html#N101ED -
Sachiko Yoshihama, Research Staff Member, IBM
Dr. Frederik De Keukelaere, Postdoctoral Researcher, IBM
Dr. Michael Steiner, Research Staff Member, IBM
Dr. Naohiko Uramoto, Research Staff Member, IBM

[3] GWT in Action: Easy Ajax with the Google Web Toolkit - Robert Hanson and Adam Tacy

[4] Google Web Toolkit Applications - Ryan Dewsbury (Paperback

[5] Head First Design Patterns (Head First) - Elisabeth Freeman, Eric Freeman, Bert Bates, and Kathy Sierra

[6] Design Patterns : Elements of reuable Object-Oriented Software. Addison-Wesley, 1995 - Gamma, Helm, Johnson, Vlissides