1. AJAX 개요

1.1 개요

AJAX는 XML에 기반 한 종합 기술로 비동기 자바스크립트 XML(Asynchronous JavaScript + XML)을 줄인 말로 자바 스크립트로 HTTP 요청을 보내서 XML 응답을 받아서 사용하는 기술이다.

즉, AJAX는 하나의 독립된 기술이 아닌 화면은 표준 HTML+CSS로 작성하고, 화면에 대한 조작과 상호작용은 문 개체 모델(Document Object Model)로, 데이터 교환은 XML 형식으로, 데이터 변환과 검색에는 XML 기술인 XSL과 XPath를, 비동기 통신에는 웹 브라우저에 내장된 XMLHttpRequest 개체를 사용하고, 이 모든 것을 하나로 묶는 언어로 자바스크립트를 사용하는 것을 AJAX라 한다.

■ 특징
- 비동기 자바스크립트 XML(Asynchronous JavaScript + XML)
- 자바 스크립트로 HTTP 요청을 보내서 XML 응답을 받아서 사용하는 기술
- HTTP 요청을 보냄 → XML 문서를 응답으로 받음 → 자동으로 XML 개체가 생성됨 → 자바스크립트는 XML 개체에 접근하여 다양한 작업을 함.
- HTML+CSS, DOM, XML, XSLT, XPath, XMLHttpRequest, JavaScript를 합쳐 사용

■ 장점
- 비동기 통신을 사용함으로써 데이터를 보내고 나서도 사용자는 다른 작업을 할 수 있다.
- 데이터만 들어가 있는 형식으로 응답을 받으므로 전통적인 웹 어플리케이션에 비해서 서버 측 처리 속도도 빠르고 전송 데이터양도 훨씬 적다.
- 응답으로 받은 XML 문서를 XML 개체로 접근하여 스크립트로 조작하고 XPath를 사용하여 XML 문서를 검색하거나 XSL을 사용하여 변환을 할 수 있다. 따라서 실행 속도가 빠르다.
- 불필요한 데이터 요청을 최소화할 수 있고, 많은 일이 클라이언트인 웹 브라우저에서 처리될 수 있다.

■ 단점
- 외부 검색 엔진이 웹페이지를 검색할 수 없는 문제가 있다.
- AJAX는 클라이언트 풀링 방식으로 실시간 서비스를 제공할 수 없다.
- AJAX가 포함된 HTML 페이지가 속한 서버가 아닌 다른 서버로 요청을 보낼 수 없고, 클라이언트 PC의 파일에 접근할 수도 없다.

■ 기존 웹 사이트에서 AJAX를 활용하면 효과가 있는 경우
- 웹 페이지를 바꾸지 않고 현재 페이지에서 어떤 동작을 하고 싶을 때
- 불필요한 팝업을 사용하여 처리하는 작업

■ AJAX 애플리케이션으로 개발할 필요가 있는 경우
- 여러 번 불필요한 화면을 다시 출력할 때
- 특정한 데이터를 반복 사용하면서 다양한 작업을 할 때

■ 주의 사항
- 뒤로 가기 버튼 사용의 어려움
  AJAX는 javascript를 사용해서 동작하기 때문에 페이지 단위의 브라우저에서 사용자 경험과 다르게 작동하는 경우가 있다. 특히 뒤로 가기 버튼이 무용지물이 되기 때문에 AJAX를 유해한 기술로 매도하기도 한다.(뒤로 가기 버튼은 웹서핑을 할 때 클릭 다음으로 많이 쓰이는 기능이다.) 하지만 이미 iframe을 사용한 해결책이 제시되어 있다. 또한 AJAX를 활용한 서비스 사용 경험이 증대될수록 이런 문제는 줄어들 것으로 보인다.
- 진행상황 파악의 어려움
  XMLHttpRequest를 통해 통신을 하는 경우 웹페이지 로딩과는 달리 사용자에게 아무런 진행 정보

가 주어지지 않는다. 그래서 아직 요청이 완료되지 않았는데 사용자가 페이지를 떠나거나 오작동할 우려가 발생하게 된다. 이 경우 사용자 요청이 처리 중에 있다는 표시를 화면에 보여 주도록 권고되고 있다. 예를 들어 Gmail에서는 중간 중간 "loading" 이라는 상태표시를 통해 사용자의 요청이 처리중임을 알려준다. 이러한 상태는 XMLHttpRequest.readyState의 상태를 통해 판단할 수 있다. 또한 이때 사용할 수 있는 공개된 이미지도 제공되고 있다.

■ 참고 사이트
・http://www.w3.org

1.2 AJAX 개체의 사용

AJAX를 사용하려면 먼저 AJAX 개체 생성 함수를 사용하여 AJAX 개체를 만들어야 한다. 인터넷 익스플로러의 경우 『new ActiveXObject("Microsoft.XMLHTTP")』를 실행하며, 모질라 웹브라우저의 경우 『new XMLHttpRequest()』를 실행하여 AJAX 개체를 생성한다. 인터넷 익스플로러 7.0 부터는 모질라 웹 브라우저 처럼 『new XMLHttpRequest()』를 통해 개체를 생성할 수 있다.

```
function createXMLHttpRequest() {
    var xmlReq = false;

    if (window.XMLHttpRequest) {     // Non-Microsoft browsers
        xmlReq = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        try {
            xmlReq = new ActiveXObject("Msxml2.XMLHTTP");
                              // XMLHttpRequest in later versions of Internet Explorer
        } catch (e1) {
            try {
                xmlReq = new ActiveXObject("Microsoft.XMLHTTP");
                              // Try version supported by older versions of Internet Explorer
            } catch (e2) {
                // Unable to create an XMLHttpRequest with ActiveX
            }
        }
    }

    return xmlReq;
}
```

window.ActiveXObject 는 ActiveXObject를 지원하는 브라우저라면 오브젝트를 리턴하고 그렇지 않다면 null를 리턴하게 된다.

■ XMLHttpRequest 객체 메서드
・void open(string method, string url, boolean asynch, string username, string password)
요청을 초기화한다. 파라미터 중에서 method, url 두개만 필수항목이고 나머지는 선택항목이다. method 는 POST, GET, PUT 3가지 중 하나를 사용하면 되며, url 은 요청하고자 하는 서버의 url 이다. asynch 는 요청이 비동기인지 여부를 판단하는 항목이다. 입력하지 않으면 디폴트로 true 가 설정되어 요청은 비동기로 처리된다. false 로 설정하면 요청은 동기로 처리되기 때문에 서버에서 응답을 받을 때까지 프로세스는 기다리게 된다. 사실 XMLHttpRequest 을 사용하는 가장 큰 이점중의 하나인 비동기 처리를 위해서는 asynch 항목을 true 로 설정해서 사용해야 한다. false 를 설정한다면 XMLHttpRequest 을 사용하는 이점이 그만큼 줄어든다.

・void send(content)
실질적으로 요청을 서버로 보낸다. 요청이 비동기이면 이 메서드는 바로 리턴 되지만 요청이 동기

이면 서버에서 응답을 받을 때 까지 계속 대기한다. content 는 선택사항이며, DOM 객체(XML 객체)이거나 input stream, string 값으로 설정할 수 있으며 HttpRequest body 의 한 부분으로 서버로 전달된다. content 에 값을 넘기려면 open() 메서드는 반드시 POST 로 설정해야 하며, GET 방식으로 요청하려면 null 을 설정하면 된다.

open(), send() 메서드가 가장 많이 사용되는 메서드들이다.

・void setRequestHeader(string header, string value)
header 에 해당하는 value 값으로 HttpRequest 헤더에 값을 설정하는 메서드로써, 반드시 open() 메서드 다음에 위치해야 한다.

・void abort()
요청을 중지한다.

・string getAllResponseHeaders()
요청에 대응되는 응답의 헤더정보를 리턴 한다. 즉, Content-Length, Date, URI 등을 포함하는 헤더정보를 string 형식으로 반환한다.

・string getResponseHeader(string header)
응답의 헤더 정보 중에서 header 에 대응되는 값을 string 형식으로 반환한다.

■ XMLHttpRequest 객체 속성
・onreadystatechange
자바스크립트 콜백 함수(funtion pointer)를 저장한다. 콜백 함수는 readyState 값이 변할 때 마다 호출된다. 요청이 서버로 보내지면 readyState 는 5가지 숫자 값으로 계속 변화가 일어나게 된다.

・readyState
요청의 상태를 의미한다.
- 0 : uninitialized (open() 호출 전)
- 1 : loading (send() 호출 전)
- 2 : loaded (http 요청 후 응답을 받은 상태)
- 3 : interactive (응답을 받고 있는 중간 상태)
- 4 : completed (모든 요청 응답 완료)

・responseText
서버의 응답을 string 형식으로 나타낸다. 단순 text를 innerHTML 속성으로 표현하기에는 알맞지만 논리적으로 파싱하거나 동적으로 페이지 컨텐츠를 생성하기는 힘들다.

・responseXML
서버의 응답을 XML 로 나타낸다. 이 속성은 DOM 객체로 파싱할 수 있다.

・status
서버로부터의 HTTP 상태코드이다.(예 200(OK), 404(NOT Found), 202(결과 값이 없을 때) 등)

・statusText
HTTP 상태코드에 대한 텍스트 값이다.(예 OK, NOT Found 등)

■ 간단한 예제 작성
1) 웹_루트₩study₩ajaxTest₩test.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
<title>Simple XMLHttpRequest</title>
<meta http-equiv='Content-Type' content='text/html; charset=euc-kr' />

<script type="text/javascript">
var xmlHttp;


function createXMLHttpRequest() {
    var xmlReq = false;

    if (window.XMLHttpRequest) {
        xmlReq = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        try {
            xmlReq = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e1) {
            try {
                xmlReq = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e2) {
                // Unable to create an XMLHttpRequest with ActiveX
            }
        }
    }

    return xmlReq;
}

function startRequest() {
    xmlHttp = createXMLHttpRequest();
    xmlHttp.onreadystatechange = handleStateChange;
    xmlHttp.open("GET", "test.xml", true);
    xmlHttp.send(null);
}

function handleStateChange() {
    if(xmlHttp.readyState == 4) {
        if(xmlHttp.status == 200) {
            alert("The server replied with: " + xmlHttp.responseText);
        }
    }
}
</script>
</head>

<body>
    <form action="#">
        <input type="button"
            value="Start Basic Asynchronous Request" onclick="startRequest();"/>
    </form>
</body>
</html>
```

2) 웹_루트₩study₩ajaxTest₩test.xml

```
Ajax Test
```

3) 실행

```
http://localhost:8080/study/ajaxTest/test.htm
```

4) 분석

(1) 사용자가 웹 브라우저 주소 칸에 "http://localhost:8080/study/ajaxTest/test.htm" 라고 입력하면 웹 브라우저는 웹 서버로 http 요청을 보낸다.

(2) 웹 서버는 요청한 html 문서를 웹 브라우저로 돌려준다.

(3) 사용자가 버튼을 클릭하면 이벤트가 발생해서 startRequest() 메서드가 실행된다.

(4) XHR 객체가 생성되고 handleStateChange() 콜백 함수가 XHR 객체의 onreadystatechange 속성에 저장된다.

(5) GET/비동기 방식으로 서버에 요청을 보내는데, 이때 XHR 객체는 서버의 test.xml 파일을 요구한다.

(6) 서버는 Ajax 클라이언트의 요청 URL 인 test.xml 을 찾아서 읽은 후에 string 형식으로 XHR 객체로 보낸다.

(7) 콜백 메서드는 XMLHttpRequest 의 state 가 변할 때 실행되므로 readyState=4 이고 stat=200 일 때 결과 값을 브라우저에 보낸다.

그리고 XMLHttpRequest 객체는 요청할 수 있는 서버의 리소스 URL 에 제한이 걸려있다. 즉, 접근할 수 있는 서버 리소스 URL 은 XMLHttpRequest 객체가 존재하는 도메인에 있어야 한다. 즉, XMLHttpRequest 객체와 서버 프로그램은 같은 도메인에 있어야만 한다. 만약 XMLHttpRequest 객체가 자기가 속해있는 도메인이 아닌 그 밖에 있는 서버의 URL 을 호출하면 브라우저 마다 차이가 있다. IE 의 경우에는 alert 창을 띄우면서 보안 위해요소가 있으니 계속 진행할 것인지 아닌지를 사용자가 판단할 수 있게 되어있고, FireFox 의 경우는 에러를 보여주며 요청자체를 브라우저에서 차단해 버린다.

Ajax 구동방식이 일반적인 웹의 방식과는 많이 다르고 또한 그 내부로직을 이해하는데 어려움이 있을 수도 있다.