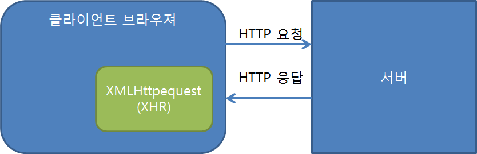
**AJAX 통신에 대하여**

2012.02.26

**김진형**

1. **XHR**
   1. **개요**
      1. ****
      2. XMLHttpRequest object는 HTTP 클라이언트 인터페이스를 구현한다. 이 인터페이스는 스크립트 엔진을 통해 노출되며 폼 데이터를 보내거나 서버로부터 데이터를 받아오는 기능을 수행한다.
      3. XMLHttpRequest라는 이름은 웹의 호환성을 위해 지어졌지만 이름의 각 부분은 잠재적으로 오해를 일으킬 수 있다. 첫째, XMLHttpRequest는 **XML을 포함한 텍스트 기반의 어떠한 포맷도 지원한다**. 둘째, XMLHttpRequest는 HTTP와 HTTPS 프로토콜 모두 사용할 수 있다. (몇몇 구현은 HTTP와 HTTPS 이외의 프로토콜을 지원하기도 하지만 그러한 기능은 본 규격에서는 다루지 않는다.). 마지막으로 XMLHttpRequest의 "request"는 HTTP에 정의된 용어에 비해 광의적으로 사용된다. 즉, HTTP method들을 위한 HTTP request와 response를 수행하기 위한 행위들을 포함한다.
      4. 또한 **크로스 도메인이 불가능 하여 제한되는** 측면이 있다.
   2. **요청방식**
      1. 클라이언트 브라우저에 내장된 XMLHttpRequest 객체를 사용하여 웹서버에 문서를 요청하는 방식이다.
         1. 크로스 도메인 : 불가능
         2. 송신방법 : GET/ POST /DELETE/ PUT
         3. 동기/비동기 : 모두가능
         4. 요청 헤더 : 가능
         5. 응답 헤더 : 가능
         6. 응답 형태 : String, XML, JSON
   3. **장단점**
      1. 장점
         1. 응답 형식을 다양하게 **String, XML, JSON** 형식 을 모두 사용할 수 있다
         2. 문서 요청 실패시 HTTP 응답 코드를 통해서 원인을 파악할 수 있다.
      2. 단점
         1. 크로스 도메인 요청이 불가능
   4. 구현 원리

|  |
| --- |
| **function** ajaxRequestXml() {  **var** oAjax = **new** $Ajax('text/data\_XML\_for\_XHR.xml',{  type : 'xhr',  method : 'get',  onload : **function**(res){  **var** list = $Element('list');  list.empty();    **var** elData = cssquery.getSingle('data',res.xml());  list.html(elData.firstChild.nodeValue);  },    });  oAjax.request();    }; |

* + 1. XMLHttpRequest 객체는 다섯 가지 상태를 가질 수 있다. **UNSENT, OPEND, HEADERS\_RECEIVED, LOADING, DONE** 이다. **현재 상태는 readystate**에 의해 노출된다.
       1. 객체가 생성되었을떄는 UNSENT 상태 이어야 한다. 상수 값은 0으로 표현된다
       2. OPENED 상태는 Open() 메서드가 성공적으로 로딩 됐을 때 이다. 이상태에서는 setRequestHeader()를 이용해 request 헤더를 설정할 수 있으며, send()를 통해 request 를 보낼 수 있다. 값은 1 이다. 또한 flag를 가지게 되는데 true, false로 나타내어 지고 초기값은 false 이다
       3. HEADERS\_RECEIVED 상태는 모든 response 헤더가 받아진 객체의 상태이다. 값은 상수 2로 표현
       4. LOADING 상태는 response entity body 가 받아지고 있는 중의 객체의 상태이다. 이 상태는 값이 3인 LOADING 상수에 의해 표현
       5. DONE 상태는 데이터의 전송이 끝나거나 전송 중 무한 리다이렉션 같은 오류가 발생한 객체의 상태 값은 4이다. 이 상태는 ERROR 플래그를 가지고 있다.

1. **JSONP**
   1. **의미와 내용**
      1. **XMLHttpRequest 가 동일 서버에만 접근할 수 있다는 한계점을 극복**하고 **다른 서버에도 동적으로 데이터를 서버에 전달하고 응답을 받아서 처리하기 위해 고안된 기술** 이다. 기본적으로 **get 방식 호출만 가능하기 때문에 많은 한계**를 가질 수 밖에 없다.
      2. 모든 data는 GET 방식으로 전달된다. 여기서 주의할 것은 \_callback 인데, 서버에서는 이 이름으로 전달받은 함수를 다음과 같이 사용해서 응답을 보내야한다. 사용 할때는 **callbackid, callbackname** 이라는 것을 지정할 수 있다.
      3. 또한 요청데이터와 응답데이터의 캐릭터 셋은 UTF-8일 것을 강제한다.
   2. **요청 방식**
      1. 크로스도메인 : 가능
      2. 송신 방법: GET
      3. 동기/비동기 : 비동기
      4. 요청헤더 : 불가능
      5. 응답 헤더 : 불가능
      6. 응답 형태 : JSON
   3. **장단점**
      1. 장점 : 크로스 도메인 요청이 가능하며, 크로스 도메인 요청이 가능한 방식 중 가장 빠른 속도를 낸다.
      2. 단점
         1. 예외 처리가 쉽지 않다
         2. POST 전송이 불가능 하다
         3. 요청한 문서에 콜백 함수를 호출하는 스크립트 작성해야 한다.
   4. **원리**

|  |
| --- |
| **function** ajaxRequestJsonpPlain() {    **var** oAjax = **new** $Ajax('text/data\_PlainText\_for\_JSONP.txt',{  type : 'jsonp',  method : 'get', //자동으로 설정됨  jsonp\_charset : 'utf-8',    onload : **function**(res){  **var** aTemp = res.json();  $Element('list').html(aTemp);  },  callbackid: '12345',  callbackname : '\_callback',  timeout : 3,  ontimeout : **function**(){  alert("Timeout!");  },    });  oAjax.request();  }; |

* + 1. 간단히 원리를 설명하면 우선 동적으로 script 태그를 생성하고 src부분에 호출할 서버 주소와 파라미터를 붙여서 넣고 document.body에 DOM메소드를 사용하여 append한다. 그렇게 하면 서버에 리퀘스트(요청)를 보내고 되고 서버는 리스판스(응답)를 하게 된다그리고 그 응답은 아래와 같은 문자열을 출력한다.
    2. 웹서버에 문서를 요청 할 떄 콜백 함수를 인수로 포함시켜 전달한다.
    3. 전달된 콜백 함수를 실행하는 스크립트를 웹 서버의 문서에 동적으로 포함시킨다.
    4. 서버에서 요청 처리한 후 스크립트를 실행하여 클라이언트의 콜백 함수를 호출한다. 이 때 처리한 결과는 JSON 형태의 데이터로 전달되며, 콜백 함수의 인수로 입력.

1. **IFRAME**
   1. **개요**
      1. Iframe을 프록시로 사용하여 웹 서버에 문서를 요청하는 방식이다. Iframe 방식을 사용하면 문서를 요청하는 쪽과 요청받는 쪽 모두 프록시용 파일이 필요하다.
   2. **요청 방식**
      1. 크로스 도메인 : 가능
      2. 송신 방법 : get
      3. 동기/비동기 : 비동기
      4. 요청 헤더 : 가능
      5. 응답 헤더 : 가능
      6. 응답 형태 : 문자열, json
   3. **장단점**
      1. 장점 : 크로스 도메인 요청이 가능하다
      2. 단점
         1. 인터넷 익스플로러에서 요청할 때마다 딱딱 효과음 발생
         2. 구조가 복잡하다
         3. Remote Proxy에서 local proxy로 get 방식으로 데이터를 전송하므로 응답 데이터의 양이 크면 문제가 발생할 수 있다.