2. 서버와 통신

2.1 responseText 속성을 이용한 단순 문자열 다루기

간단하고 단순한 문자열을 처리하는데 적합한 XMLHttpRequest 객체의 resonseText 속성만을 이용하여 문자열을 처리하는 방법의 예제를 작성하면,

■ resonseText 속성 예제

1) 웹_루트₩study₩ajaxTest₩innerHTML.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>Using responseText with innerHTML</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", "innerHTML.xml", true);
    xmlHttp.send(null);
}

function handleStateChange() {
    if(xmlHttp.readyState == 4) {
        if(xmlHttp.status == 200) {
            document.getElementById("results").innerHTML = xmlHttp.responseText;
```

```
        }
    }
}
</script>
</head>

<body>
    <form action="#">
        <input type="button" value="눌러봐 ..." onclick="startRequest();"/>
    </form>
    <div id="results"></div>
</body>
</html>
```

2) 웹_루트₩study₩ajaxTest₩innerHTML.xml

```
<table border='1'>
    <tbody>
        <tr>
            <th>색상</th>
            <th>RGB</th>
        </tr>
        <tr>
            <td>검정</td>
            <td>#000000</td>
        </tr>
        <tr>
            <td>파랑</td>
            <td>#0000FF</td>
        </tr>
        <tr>
            <td>빨강</td>
            <td>#FF0000</td>
        </tr>
    </tbody>
</table>
```

3) 실행

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

2.2 responseXML 속성을 이용한 DOM 시작하기

복잡한 응답데이터의 경우는 단순한 문자열로 처리할 수 없으며 XML 형식으로 처리하는 것이 훨씬 논리적이고 효율적이며 브라우저는 서버로부터 받은 XML 형식의 데이터를 W3C 의 DOM 을 이용해서 처리된다. 즉, DOM 을 지원하는 브라우저들은 XML 문서를 다루는 많은 API 를 구현하고 있다.

DOM 은 HTML 과 XML 을 다루는 API 를 제공하고 있으며, 스크립트를 통해서 도큐먼트에 접근할 수 있도록 정의되어 있다. 자바스크립트는 DOM 에 접근할 수 있고 DOM 을 다룰 수 있는 스크립팅 언어이다. 다큐먼트의 모든 요소들은 DOM 의 부분들이기 때문에 요소의 속성과 메서드들은 자바스크립트로 제어가 가능하다.

■ [참고] - DOM(Document Object Model)
프로그램과 스크립트에서 문서의 내용, 구조 및 스타일에 동적으로 액세스하고 업데이트할 수 있도

록 하는 플랫폼 중립 및 언어 중립 인터페이스로 문서 개체 모델은 HTML 및 XML 문서를 나타내기 위한 표준 개체 집합, 이 개체를 결합하는 방법의 표준 모델 및 이 모델에 액세스하고 조작하기 위한 표준 인터페이스를 제공한다. 공급업체에서는 소유 데이터 구조 및 API에 대한 인터페이스로 DOM을 지원할 수 있으며 내용 작성자는 제품별 API 대신 표준 DOM 인터페이스에 쓸 수 있으므로 웹에서 상호 운용성을 향상시킬 수 있다.

■ XML 문서를 처리하기 위한 DOM 요소의 속성
- childNodes : 현재 요소의 자식을 배열로 표현한다.
- firstChild : 현재 요소의 첫 번째 자식이다.
- lastChild : 현재 요소의 마지막 자식이다.
- nextSibling : 현재 요소와 바로 다음의 요소를 의미한다.
- nodeValue : 해당 요소의 값을 읽고 쓸 수 있는 속성을 정의한다.(=data)
- parentNode : 해당 요소의 부모 노드이다.
- previousSibling : 현재 요소와 바로 이전의 요소를 의미한다.

■ XML 도큐먼트를 다루는 유용한 DOM 요소의 메서드
- getElementById(id) : 도큐먼트에서 특정한 id 속성 값을 가지고 있는 요소를 반환한다.
- getElementsByTagName(name) : 특정한 태그 이름을 가지고 있는 자식 요소로 구성된 배열을 리턴 한다.
- hasChildNodes() : 해당 요소가 자식 요소를 포함하고 있는지를 나타내는 Boolean 값을 리턴 한다.
- getAttribute(name) : 특정한 name 에 해당하는 요소의 속성 값을 리턴 한다.

■ 예제

1) 웹_루트₩study₩ajaxTest₩parseXML.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>Parsing XML Responses with the W3C DOM</title>
<meta http-equiv='Content-Type' content='text/html; charset=euc-kr'/>

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


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(requestedList) {
    requestType = requestedList;
    xmlHttp = createXMLHttpRequest();
    xmlHttp.onreadystatechange = handleStateChange;
    xmlHttp.open("GET", "parseXML.xml", true);
    xmlHttp.send(null);
}

function handleStateChange() {
    if(xmlHttp.readyState == 4) {
        if(xmlHttp.status == 200) {
            if(requestType == "seoul") {
                listSeoulCity();
            }
            else if(requestType == "all") {
                listAllCity();
            }
        }
    }
}

function listSeoulCity() {
    var xmlDoc = xmlHttp.responseXML;
    var seoulNode = xmlDoc.getElementsByTagName("서울")[0];

    var seoulCity = seoulNode.getElementsByTagName("구");

    outputList("서울", seoulCity);
}

function listAllCity() {
    var xmlDoc = xmlHttp.responseXML;
    var allCity = xmlDoc.getElementsByTagName("구");

    outputList("전체", allCity);
}

function outputList(title, city) {
    var out = title;
    var currentCity = null;
    for(var i = 0; i < city.length; i++) {
        currentCity = city[i];
        out = out + "₩n- " + currentCity.childNodes[0].nodeValue;
    }

    alert(out);
}
</script>
</head>

<body>
```

```
    <h1>Process XML Document</h1>
    <br/><br/>
    <form action="#">
        <input type="button" value="전체" onclick="startRequest('all');"/>
        <br/><br/>
        <input type="button" value="서울" onclick="startRequest('seoul');"/>
    </form>
</body>
</html>
```

2) 웹_루트₩study₩ajaxTest₩parseXML.xml

```
<?xml version="1.0" encoding="euc-kr"?>
<도시>
    <서울>
        <구>강남구</구>
        <구>영등포구</구>
        <구>종로구</구>
    </서울>
    <인천>
        <구>부평</구>
        <구>남동</구>
        <구>연수</구>
    </인천>
</도시>
```

3) 분석

(1) listAllCity() 메서드

var xmlDoc = xmlHttp.responseXML;

→ XHR 객체는 responseXML 속성을 이용해서 서버로부터의 XML 결과 도큐먼트를 다룰 수 있는 DOM 속성 및 메서드를 사용할 수 있게 해준다.

var allCity = xmlDoc.getElementsByTagName("구");

→ XML 결과 도큐먼트로 부터 "구" 자식 엘리먼트들로 구성된 배열을 얻어와 allCity 변수에 할당하는 로직이다.

(2) listSeoulCity() 메서드

var seoulNode = xmlDoc.getElementsByTagName("서울")[0];

→ XML 다큐먼트에서 "서울" 앨리먼트는 유일하게 하나만 존재하므로 자식 앨리먼트로 구성된 배열 중에서 첫 번째(0) 배열 값을 얻어 와야 한다. 위 식은 아래와 같이 수정해도 결과는 같다.

var seoulNode = xmlDoc.getElementsByTagName("서울").item(0);

(3) outputList() 메서드

out = out + "₩n- " + currentState.childNodes[0].nodeValue;

→ 이 부분은 각각의 "구" 앨리먼트의 첫 번째 자식 노드의 값을 out 변수에 계속 연결하는 부분이다. "구" 앨리먼트의 값을 표현하고 있는 부분도 XML 에서는 하나의 text 엘리먼트이다. 따라서 각각의 "구" 엘리먼트의 첫 번째 text 자식 엘리먼트를 childNodes[0] 으로 표시한 것이며 그 값을 가져오기 위해서 nodeValue 속성이 사용된 것이다. nodeValue 는 아래와 같이 data 속성을 사용해도 같은 결과를 얻는다.

out = out + "₩n- " + currentState.childNodes[0].data;

2.3 Dynamic DOM 객체 다루기

위에서 다뤄본 DOM 의 기초적인 속성 및 메서드들은 다이나믹 한 웹페이지를 구성하는데 한계가 있다. 웹페이지 전체가 리로딩 되지 않고 적절한 시점에 필요한 부분만 서버와 통신하여 데이터가 수정되는 동적인 웹페이지를 만들려면 더 다양한 DOM 의 속성을 익혀야 한다.

따라서 컨텐츠를 동적으로 생성할 수 있게 해주는 W3C DOM 의 속성과 메서드에는 어떤 것 들이 있는지 알아보면,

■ 메서드 및 속성

- document.createElement(tagName)
  tagName 으로 된 엘리먼트를 생성한다. div 를 메서드 파라미터로 입력하면 div 엘리먼트가 생성된다.
- document.createTextNode(text)
  정적 텍스트를 담고 있는 노드를 생성한다.
- <element>.appendChild(childNode)
  특정 노드를 현재 엘리먼트의 자식 노드에 추가시킨다.(예를 들어 select 엘리먼트에 option 엘리먼트 추가)
- <element>.getAttribute(name)
  속성명이 name 인 속성 값을 반환한다.
- <element>.setAttribute(name, value)
  속성 값 value 를 속성명이 name 인 곳에 저장한다.
- <element>.insertBefore(newNode, tartgetNode)
  newNode 를 tartgetNode 전에 삽입한다.
- <element>.removeAttribute(name)
  엘리먼트에서 name 속성을 제거한다.
- <element>.removeChild(childNode)
  자식 엘리먼트를 제거한다.
- <element>.replaceChild(newNode, oldNode)
  oldNode 를 newNode 로 치환한다.
- <element>.hasChildNodes()
  자식 노드가 존재하는지 여부를 판단한다. 리턴형식은 Boolean 이다.

현재 거의 모든 브라우저는 DOM 을 지원하고 있으며 API 또한 비슷하게 동작하도록 구현되어 있다. 즉, DOM API 의 구현이 브라우저마다 다소 차이가 있으며 가장 호환이 안 되는 브라우저는 인터넷 익스플로러이다.

■ IE 에서 문제가 되고 있는 DOM 객체의 특징

- IE 에서는 <table> 에 <tr> 을 추가할 때 appendChilde() 메서드를 사용하더라도 <tr> 은 나타나지 않는다. 따라서 <tr> 을 <tbody> 에 추가해 주는 방식을 사용해야 모든 브라우저에서 동작하는 코드를 작성할 수 있다.
- IE 에서는 setAttribute() 메서드에 class 속성을 이용할 수 없다. setAttribute("class", "newClassName") 한 후에 다시 setAttribute("className", "newClassName") 을 해야 모든 브라우저의 호환성을 보장할 수 있다.
- IE 에서는 style 속성에 setAttribute() 메서드를 이용할 수 없다. <element>.setAttribute("style, font-weight:bold;") 라고 하는 대신에 <element>.style.cssText = "font-weight:bold" 라고 해야 모든 브라우저에서 제대로 작동한다.

■ 예제

1) 웹_루트₩study₩ajaxTest₩dynamicContent.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>Dynamically Editing Page Content</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 doSearch() {
    xmlHttp = createXMLHttpRequest();
    xmlHttp.onreadystatechange = handleStateChange;
    xmlHttp.open("GET", "dynamicContent.xml", true);
    xmlHttp.send(null);
}

function handleStateChange() {
    if(xmlHttp.readyState == 4) {
        if(xmlHttp.status == 200) {
            clearPreviousResults();
            parseResults();
        }
    }
}

function clearPreviousResults() {
    var header = document.getElementById("header");
    if(header.hasChildNodes()) {
        header.removeChild(header.childNodes[0]);
    }

    var tableBody = document.getElementById("resultsBody");
    while(tableBody.childNodes.length > 0) {
        tableBody.removeChild(tableBody.childNodes[0]);
    }
}
```

```
function parseResults() {
    var results = xmlHttp.responseXML;

    var property = null;
    var book = "";
    var price = "";
    var comments = "";

    var properties = results.getElementsByTagName("property");
    for(var i = 0; i < properties.length; i++) {
        property = properties[i];
        book = property.getElementsByTagName("book")[0].firstChild.nodeValue;
        price = property.getElementsByTagName("price")[0].firstChild.nodeValue;
        comments = property.getElementsByTagName("comments")[0].firstChild.nodeValue;

        addTableRow(book, price, comments);
    }

    var header = document.createElement("h2");
    var headerText = document.createTextNode("Results:");
    header.appendChild(headerText);
    document.getElementById("header").appendChild(header);

    document.getElementById("resultsTable").setAttribute("border", "1");
}

function addTableRow(book, price, comments) {
    var row = document.createElement("tr");
    var cell = createCellWithText(book);
    row.appendChild(cell);

    cell = createCellWithText(price);
    row.appendChild(cell);

    cell = createCellWithText(comments);
    row.appendChild(cell);

    document.getElementById("resultsBody").appendChild(row);
}

function createCellWithText(text) {
    var cell = document.createElement("td");
    var textNode = document.createTextNode(text);
    cell.appendChild(textNode);

    return cell;
}
</script>
</head>

<body>
  <h1>결과</h1>

  <form action="#">
```

```
    <input type="button" value="출력" onclick="doSearch();"/>
  </form>

  <span id="header">
  </span>

  <table id="resultsTable" width="75%" border="0">
    <tbody id="resultsBody">
    </tbody>
  </table>
</body>
</html>
```

2) 웹루트₩study₩ajaxTest₩dynamicContent.xml

```
<?xml version="1.0" encoding="euc-kr"?>
<properties>
    <property>
        <book>JAVA</book>
        <price>100,000</price>
        <comments>자바</comments>
    </property>
    <property>
        <book>XML</book>
        <price>110,000</price>
        <comments>그냥</comments>
    </property>
    <property>
        <book>JSP</book>
        <price>90,000</price>
        <comments>ㅋㅋ</comments>
    </property>
</properties>
```

위 예제에서 중요한 부분을 정리하면,

```
function createCellWithText(text) {
    var cell = document.createElement("td");
    var textNode = document.createTextNode(text);
    cell.appendChild(textNode);

    return cell;
}
```

위 메서드는 테이블 컬럼(<td></td>)에 해당하는 정보를 생성하는 메서드이다. 하나의 row 에는 3 개의 컬럼 요소가 있으며 동적으로 하나의 행을 생성하기 위해서는 book, price, comments 에 해당하는 td 요소를 각각 생성해야 한다.

```
function addTableRow(book, price, comments) {
    var row = document.createElement("tr");
    var cell = createCellWithText(book);
    row.appendChild(cell);

    cell = createCellWithText(price);
    row.appendChild(cell);
```

```
        cell = createCellWithText(comments);
        row.appendChild(cell);

        document.getElementById("resultsBody").appendChild(row);
    }
```

위 메서드는 테이블 행(<tr></tr>)에 해당하는 정보를 생성해서 테이블에 추가하는 메서드이다. 행을 구성 하고 있는 3개의 컬럼 요소를 각각 만들어서 row 변수에 추가한 후, 이 변수를 다시 tbody 속성에 추가 표를 완성하게 된다.

2.4 요청 파라미터를 서버로 보내기

XMLHttpRequest(XHR) 은 고전적 웹의 GET/POST 방식과 흡사하게 동작하며 GET 방식은 name=value 쌍의 파라미터를 URL 에 실어서 서버로 전송한다. name=value 쌍은 리소스 url 의 끝을 의미하는 ? 이후에 구분자(&) 를 둔다.

POST 방식은 GET 방식과 마찬가지로 name=value 쌍의 형태로 데이터를 전달하며 구분자 (&)를 사용한다. 하지만 POST 방식은 폼 요소의 데이터를 인코딩하여 Http Request 객체의 body 에 저장해서 보낸다.

또 다른 차이점이 있다면 서버로 보낼 수 있는 요청정보의 크기인데, GET 방식으로는 name1=value1&name2=value2&name3=value3... 이런 문자열의 길이가, 브라우저마다 차이가 있지만, 대략 2000 byte 이상이면 불가능하다. 따라서 서버로 보내는 파라미터가 많을 때는 POST 방식을 사용해야 한다. 일반적으로 데이터를 fetch(검색) 할 때는 GET 방식을 사용하고 그 이외의 작업(추가, 수정, 삭제)에는 POST 방식을 주로 사용한다.

Ajax와 관련된 차이점이라면 GET 방식은 파라미터가 인코딩되어 url 에 붙어가기 때문에 해당 url 전체를 재사용(bookmark) 가능하지만 Ajax 특성상 이런 북 마킹 기능은 불가능 하다. HTML 폼 요소에는 method 속성이 있는데 개발자는 GET 또는 POST 방식을 선택할 수 있다. 요청 데이터들은 서버로 submit 될 때 method 속성에 알맞도록 자동으로 인코딩되지만 XHR 객체는 이런 내장 알고리즘이 없기 때문에 개발자가 쿼리 스트링을 작성해야 한다. 쿼리 스트링을 작성하는 방법은 GET 또는 POST 방식에 상관없이 동일하다. 유일한 차이점이 있다면 GET 방식의 쿼리 스트링은 요청 url 에 붙어서 서버로 전송되지만 POST 방식의 쿼리 스트링은 XHR 객체의 send(쿼리 스트링) 메서드가 호출될 때 파라미터로 전송된다.

■ 예제

1) 웹_루트₩study₩ajaxTest₩getAndPostExam.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>Sending Request Data Using GET and POST</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 createQueryString() {
    var name = document.getElementById("name").value;
    var age = document.getElementById("age").value;
    var birthday = document.getElementById("birthday").value;

    var queryString = "name=" + encodeURIComponent(name) + "&age=" + age
        + "&birthday=" + birthday;

    return queryString;
}

function doRequestUsingGET() {
    xmlHttp = createXMLHttpRequest();

    var queryString = "getAndPostExam.jsp?";
    queryString = queryString + createQueryString()
        + "&timeStamp=" + new Date().getTime();
    xmlHttp.onreadystatechange = handleStateChange;
    xmlHttp.open("GET", queryString, true);
    xmlHttp.send(null);
}

function doRequestUsingPOST() {
    createXMLHttpRequest();

    var url = "getAndPostExam.jsp?timeStamp=" + new Date().getTime();
    var queryString = createQueryString();

    xmlHttp.open("POST", url, true);
    xmlHttp.onreadystatechange = handleStateChange;
    xmlHttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    xmlHttp.send(queryString);
}

function handleStateChange() {
    if(xmlHttp.readyState == 4) {
        if(xmlHttp.status == 200) {
```

```
            parseResults();
        }
    }
}

function parseResults() {
    var responseDiv = document.getElementById("serverResponse");
    if(responseDiv.hasChildNodes()) {
        responseDiv.removeChild(responseDiv.childNodes[0]);
    }

    var responseText = document.createTextNode(xmlHttp.responseText);
    responseDiv.appendChild(responseText);
}
</script>
</head>
<body>
  <h1>Enter your name, age, and birthday:</h1>

  <table>
    <tbody>
        <tr>
            <td>이름:</td>
            <td><input type="text" id="name"/>
        </tr>
        <tr>
            <td>나이:</td>
            <td><input type="text" id="age"/>
        </tr>
        <tr>
            <td>생년월일:</td>
            <td><input type="text" id="birthday"/>
        </tr>
    </tbody>

  </table>

  <form action="#">
    <input type="button" value="전송 [GET]" onclick="doRequestUsingGET();"/>

    <br/><br/>
    <input type="button" value="전송 [POST]" onclick="doRequestUsingPOST();"/>
  </form>
  <br/>
  <h2>Server Response:</h2>
  <div id="serverResponse"></div>
</body>
</html>
```

2) 웹_루트₩study₩ajaxTest₩getAndPostExam.jsp

```
<%-- JSP 페이지 정의 시작 --%>
<%@ page contentType = "text/html; charset=euc-kr" %>
<%-- JSP 페이지 정의 끝 --%>
```

```
<%-- 문자 셋 --%>
<% request.setCharacterEncoding("UTF-8"); //Ajax 에서는 반드시 %>

<%-- JSP 코드 시작 --%>
<%
        String name = request.getParameter("name");
        String age = request.getParameter("age");
        String birthday = request.getParameter("birthday");

        if(request.getMethod().equals("GET"))
          name = new String(name.getBytes("8859_1"),"UTF-8");

        //Create the response text
        String responseText = "Hello " + name
                + ". Your age is " + age
                + ". Your birthday is " + birthday + ".";

        // Write the response back to the browser
        out.println(responseText);
%>
<%-- JSP 코드 끝 --%>
```

3) 주의사항

(1) AJAX로 요청할 때 URI의 일부로 들어가 전달되는 값은 FORM 입력처럼 특수 문자나 한글이 깨지지 않도록 인코딩해야 한다. 자바스크립트에서 escape()함수와 encodeURIComponent() 함수를 지원해 주는데 escape()은 유니코드로, encodeURIComponent는 utf-8로 인코딩 한다.

AJAX에서는 문자열을 URI 형태로 인코딩하려면 encodeURIComponent() 함수를 사용해야 하며 POST 방식에서는 문제가 없지만 GET 방식에서는 전달받은 곳에서 "8859_1" 코드를 "UTF-8"로 변환하는 작업이 필요 하다.

(2) AJAX에서는 EUC-KR 을 사용하는 것은 무리가 있는 또한 GET 방식 또는 POST 방식에 의해 전달 받은 서버 스크립트 페이지에서는 반드시 다음과 같이 request 문자 셋을 UTF-8 로 지정해야 한다.

```
<% request.setCharacterEncoding("UTF-8"); %>
```

(3) POST 방식에서는 setRequestHeader() 메서드로 Content-Type 요청 헤더의 값을 "application/x-www-form-urlencoded"로 지정해야 한다. 그렇지 않으면 파라미터를 서버에 전송하지 않는다.

폼 데이터는 웹 서버에게 전송되기 전에 웹 브라우저에 의해 일정한 형식으로 인코딩되는데, application/x-www-form-urlencoded은 URL인코딩이라고 부르는 방식이다.

(4) 쿼리 스트링을 만들 때 "&timeStamp=" + new Date().getTime(); 부분을 넣은 이유 몇몇 브라우저들은 경우에 따라 똑같은 url 로 XHR 멀티 요청을 보냈을 때 서버의 결과를 캐싱 하는 경향을 보인다. 같은 url 이지만 응답이 다를 경우엔 캐싱 하는 경향이 오히려 예상치 못한 결과의 원인이 될 수도 있기 때문에 되도록이면 unique 한 url을 생성하기 위해 사용 한 것이다.