



20 XMLHttpRequest

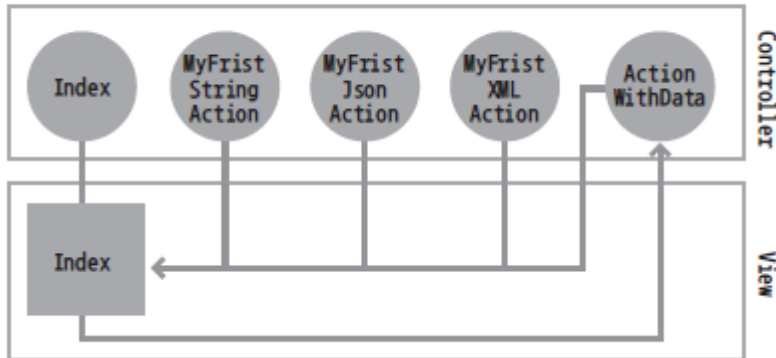
모던 웹을 위한 Javascript jQuery 입문

20.1 프로젝트 정리

❖ 기본적인 자바스크립트만으로 Ajax하는 방법

■ 현재 프로젝트 구성

그림 20-1 프로젝트 구성



❖ XMLHttpRequest

- 자바스크립트가 Ajax를 사용할 때 사용하는 객체
- XHR
- 인터넷 익스플로러 6 이하는 XMLHttpRequest 객체 없음

❖ Script 태그 생성

코드 20-1 Index.cshtml

```
@{ Layout = null; }  
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Index</title>  
    <script>  
  
    </script>  
  </head>  
  <body>  
  
  </body>  
</html>
```



❖ XMLHttpRequest 객체 사용

- XMLHttpRequest 객체부터 생성
- open() 메서드를 사용하여 편지지 전송 위치와 방식 지정
 - XMLHttpRequest.open(Method, URL, isAsync)
 - 첫 번째 매개 변수에는 GET이나 POST 같은 전송 방식 입력
 - 두 번째 매개 변수에는 요청을 수행 위치 지정

코드 20-2 Index.cshtml: XMLHttpRequest 객체 생성

```
<script>
    var request = new XMLHttpRequest();
</script>
```

코드 20-3 Index.cshtml: XMLHttpRequest 객체의 open() 메서드

```
<script>
    var request = new XMLHttpRequest();
    request.open('GET', '/Home/MyFirstStringAction', false);
</script>
```



20.2 XMLHttpRequest 객체를 사용한 Ajax 요청

❖ 예제 코드 20-4]

- send() 메서드와.responseText 속성 사용해 Ajax 수행
- Ajax 응답 출력

코드 20-4 Index.cshtml: XMLHttpRequest 객체를 사용한 기본적인 Ajax

```
<script>
    // XMLHttpRequest 객체를 생성합니다.
    var request = new XMLHttpRequest();
    request.open('GET', '/Home/MyFirstStringAction', false);

    // Ajax를 수행합니다.
    request.send();

    // 출력합니다.
    alert(request.responseText);
</script>
```

그림 20-2 출력된 request.responseText



<h1>Hello ASP.NET MVC</h1>



20.2 XMLHttpRequest 객체를 사용한 Ajax 요청

❖ 액션 ActionWithData

- name과 age의 두 가지 요청 매개 변수를 넘겨주면 문자열을 만듦

코드 20-5 Index.cshtml: XMLHttpRequest 객체를 사용한 요청 매개 변수 전달

```
<script>
    window.onload = function () {
        // XMLHttpRequest 객체를 생성합니다.
        var request = new XMLHttpRequest();
        request.open('GET', '/Home/ActionWithData?name=RintlanTta&age=21', false);

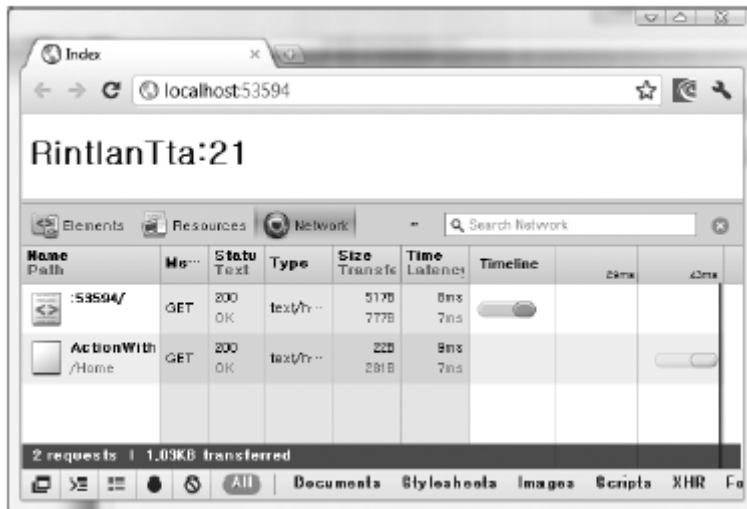
        // Ajax를 수행합니다.
        request.send();

        // 출력합니다.
        document.body.innerHTML = request.responseText;
    };
</script>
```

그림 20-3 실행 결과

RintlanTta:21

그림 20-4 요소 검사를 통한 Ajax 요청 확인



20.3 XMLHttpRequest 객체 생성

❖ 인터넷 익스플로러 6 이하의 경우

- XMLHttpRequest 객체 없지만, ActiveX 객체의 형태로 같은 기능 가능
- 예제 코드 20-6]
 - 브라우저의 종류에 따라 적절한 요청 객체 생성하는 함수 작성
 - createRequest() 함수 생성

코드 20-6 Index.cshtml: createRequest() 함수 생성

```
<script>
  // XMLHttpRequest 객체를 생성하는 함수
  function createRequest() {

  }

  window.onload = function () {
    // XMLHttpRequest 객체를 생성합니다.
    var request = new XMLHttpRequest();
    request.open('GET', '/Home/ActionWithData', false);

    // Ajax를 수행합니다.
    request.send('name=RintianTta&age=21');

    // 출력합니다.
    document.body.innerHTML = request.responseText;
  };
</script>
```



20.3 XMLHttpRequest 객체 생성

❖ createRequest() 함수

- 인터넷 익스플로러 6 이하에서는 예외 발생
 - ActiveXObject() 생성자 함수를 사용해 요청 객체 생성
 - Msxml2.XMLHTTP 또는 Microsoft.XMLHTTP 사용

코드 20-7 Index.cshtml: createRequest() 함수

```
// XMLHttpRequest 객체를 생성하는 함수
function createRequest() {
    var request;
    try {
        request = new XMLHttpRequest();
    } catch (exception) {
        try {
            request = new ActiveXObject('Msxml2.XMLHTTP');
        } catch (innerException) {
            request = new ActiveXObject('Microsoft.XMLHTTP');
        }
    }
    return request;
}
```



20.3 XMLHttpRequest 객체 생성

❖ Microsoft.XMLHTTP 객체 생성 예외 처리 추가

코드 20-8 Index.cshtml: 모든 브라우저에서 동작하는 Ajax 요청

```
<script>
// XMLHttpRequest 객체를 생성하는 함수
function createRequest() {
    var request;
    try {
        request = new XMLHttpRequest();
    } catch (exception) {
        try {
            request = new ActiveXObject('Msxml2.XMLHTTP');
        } catch (innerException) {
            request = new ActiveXObject('Microsoft.XMLHTTP');
        }
    }
    return request;
}

window.onload = function () {
    // XMLHttpRequest 객체를 생성합니다.
    var request = createRequest();
    request.open('GET', '/Home/ActionWithData', false);

    // Ajax를 수행합니다.
    request.send('name=RintlanTtagage=21');

    // 출력합니다.
    document.body.innerHTML = request.responseText;
};
</script>
```



❖ 동기 방식

- 데이터를 서버와 클라이언트가 같은 속도로 연계해 동작하는 방식
- 지금까지 예제에서 사용한 방법

❖ 비동기 방식

- XMLHttpRequest 객체 편지를 우체통에 넣고 답장이 올 때까지 다른 일 처리하는 방식

❖ 예제 코드 20-9] - send() 메서드 소비 시간 측정

코드 20-9 Index.cshtml: 동기 방식일 때 send() 메서드에 소비되는 시간 측정

```
window.onload = function () {  
    // XMLHttpRequest 객체를 생성합니다.  
    var request = createRequest();  
    request.open('GET', '/Home/MyFirstStringAction', false);  
  
    // send() 메서드에 소비되는 시간 측정  
    var prevDate = new Date();  
    request.send();  
    var nowDate = new Date();  
  
    // 출력합니다.  
    alert(nowDate - prevDate);  
};
```



20.4 동기 방식과 비동기 방식

❖ open() 메서드의 세 번째 매개 변수를 true로 바꾸고 실행

■ 비동기 방식으로 실행

- request.open('GET', '/Home/MyFirstStringAction', true);
- 데이터를 받는 동안에도 코드를 지속적 실행

– send() 메서드에서 0밀리초에서 1밀리초 이상 걸리지 않음

❖ 예제 코드 20-10]

- onreadystatechange 이벤트에 이벤트 핸들러 연결
 - XMLHttpRequest 객체의 readyState 속성 출력

코드 20-10 Index.cshtml: XMLHttpRequest 객체의 onreadystatechange 이벤트

```
window.onload = function () {  
    // XMLHttpRequest 객체를 생성합니다.  
    var request = createRequest();  
    request.onreadystatechange = function (event) {  
        // 출력합니다.  
        alert(request.readyState);  
    };  
    request.open('GET', '/Home/MyFirstStringAction', true);  
    request.send();  
};
```



❖ XMLHttpRequest 객체의 readyState 속성

표 20-1 readyState 속성값

readyState 속성	설명
0	Request 객체를 만들었지만 open() 메서드를 사용해 초기화하지 않았음
1	Request 객체를 만들고 초기화했지만 send() 메서드가 사용되지 않음
2	send() 메서드를 사용했지만 아직 데이터를 받지 못함
3	데이터의 일부만을 받음
4	모든 데이터를 받음



20.4 동기 방식과 비동기 방식

❖ 예제 코드 20-11]

- Ajax 사용해 데이터 전송 받는 시점은 readyState 속성이 4일 때
 - 이때 문서 객체와 관련된 처리

코드 20-11 Index.cshtml: 비동기 방식을 사용한 Ajax 요청 (1)

```
window.onload = function () {  
    // XMLHttpRequest 객체를 생성합니다.  
    var request = createRequest();  
    request.onreadystatechange = function (event) {  
        if (request.readyState == 4) {  
            document.body.innerHTML = request.responseText;  
        }  
    };  
  
    request.open('GET', '/Home/MyFirstStringAction', true);  
    request.send();  
};
```



❖ 예제 코드 20-12]

- 데이터를 가져왔는데 올바른 데이터가 아닌 경우 처리
 - XMLHttpRequest 객체의 status 속성이 200일 때 원하는 코드 처리

표 20-2 HTTP Status Code의 예

HTTP Status Code	설명	예
1XX	처리 중	100 Continue
2XX	성공	200 OK
3XX	리다이렉트	300 Multiple Choices
4XX	클라이언트 오류	400 Bad Request
5XX	서버 오류	500 Internal Server Error

코드 20-12 Index.cshtml: 비동기 방식을 사용한 Ajax 요청 (2)

```
window.onload = function () {  
    // XMLHttpRequest 객체를 생성합니다.  
    var request = createRequest();  
    request.onreadystatechange = function (event) {  
        if (request.readyState == 4) {  
            if (request.status == 200) {  
                document.body.innerHTML = request.responseText;  
            }  
        }  
    };  
  
    request.open('GET', '/Home/MyFirstStringAction', true);  
    request.send();  
};
```



❖ Ajax를 사용해 JSON을 가져와 다루는 방법

- XMLHttpRequest 객체를 사용해 Ajax 요청 수행
- 응답 받은 JSON을 자바스크립트 객체로 변환

코드 20-13 Index.cshhtml: JSON을 문자열을 자바스크립트 객체로 변환

```
window.onload = function () {  
    // XMLHttpRequest 객체를 생성합니다.  
    var request = createRequest();  
    request.onreadystatechange = function (event) {  
        if (request.readyState == 4) {  
            if (request.status == 200) {  
                var json = eval('(' + request.responseText + ')');  
            }  
        }  
    };  
  
    request.open('GET', '/Home/MyFirstJSONAction', true);  
    request.send();  
};
```



❖ JSON 다루기

- XMLHttpRequest 객체의 responseText 속성
 - eval() 함수의 매개 변수에 괄호로 감싸 넣음 – 에러 방지

코드 20-14 Index.cshtml: Ajax를 사용한 JSON 조작

```
request.onreadystatechange = function (event) {  
    if (request.readyState == 4) {  
        if (request.status == 200) {  
            var json = eval('(' + request.responseText + ')');  
  
            var willIn = '';  
  
            for (var i in json) {  
                willIn += '<h1>' + i + ':' + json[i] + '</h1>';  
            }  
            // 출력합니다.  
            document.body.innerHTML = willIn;  
        }  
    }  
};
```

그림 20-5 실행 결과

name:윤인성

gender:남자

part:세컨드기타



❖ 예제 코드 20-15]

- request 객체의 URL을 액션 MyFirstXMLAction으로 설정

코드 20-15 index.cshtml:액션 설정

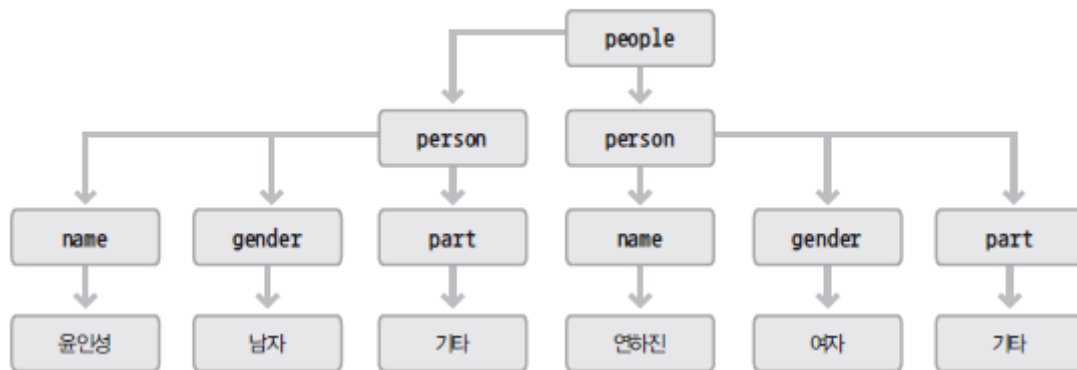
```
window.onload = function () {  
    // XMLHttpRequest 객체를 생성합니다.  
    var request = createRequest();  
  
    // onreadystatechange 이벤트 연결  
    request.onreadystatechange = function (event) {  
        if (request.readyState == 4) {  
            if (request.status == 200) {  
                alert(request.responseXML);  
            }  
        }  
    };  
  
    // XMLHttpRequest 요청  
    request.open('GET', '/Home/MyFirstXMLAction', true);  
    request.send();  
};
```



❖ 19장에서 액션 MyFirstXMLAction으로 만든 XML 문서

```
<people>
  <person>
    <name>윤인성</name>
    <gender>남자</gender>
    <part>기타</part>
  </person>
  <person>
    <name>연하진</name>
    <gender>여자</gender>
    <part>기타</part>
  </person>
</people>
```

그림 20-6 XML 문서의 DOM 표현



❖ 문서 객체를 조작할 때 사용하는 속성

표 20-3 DOM 속성

속성	설명
innerHTML	문서 객체의 내부 글자
childNodes	문서 객체의 자식 노드
attributes	문서 객체의 속성

❖ XML 문서 객체를 조작할 때 사용하는 메서드

- XML 문서는 id 속성을 사용하지 않으므로 `getElementsByTagName()` 메서드 더 많이 사용

표 20-4 DOM 메서드

메서드	설명
<code>getElementById(id)</code>	id 속성이 일치하는 문서 객체를 선택합니다.
<code>getElementsByTagName(name)</code>	태그 이름이 일치하는 문서 객체를 선택합니다.



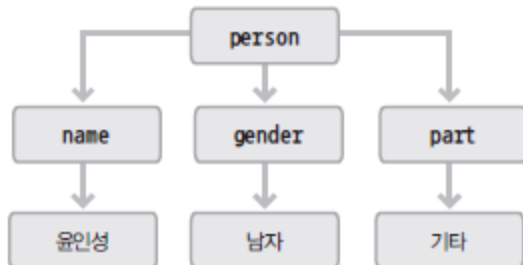
❖ 예제 코드 20-16]

- getElementByTagName() 메서드 사용
- XML 문서에서 person 태그 가져옴
 - 문서 객체를 배열 형태로 가져옴

코드 20-16 index.cshhtml

```
if (request.status == 200) {  
    // 변수를 선언합니다.  
    var xml = request.responseXML;  
  
    // XML에서 각각의 요소를 뽑아냅니다.  
    var people = xml.getElementsByTagName('person');  
    for (var i = 0; i < people.length; i++) {  
  
    }  
}
```

그림 20-7 people[0]의 형태



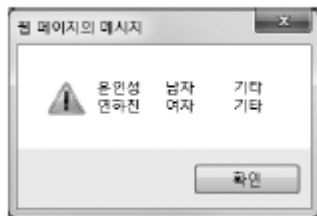
❖ 예제 코드 20-17]

■ 원하는 데이터 추출

코드 20-17 index.cshtml

```
if (request.readyState == 4) {  
    if (request.status == 200) {  
        // 변수를 선언합니다.  
        var xml = request.responseXML;  
  
        // XML에서 각 요소를 뽑아냅니다.  
        var people = xml.getElementsByTagName('person');  
        var output = '';  
        for (var i = 0; i < people.length; i++) {  
            try {  
                output += people[i].childNodes[1].childNodes[0].nodeValue + '\t';  
                output += people[i].childNodes[3].childNodes[0].nodeValue + '\t';  
                output += people[i].childNodes[5].childNodes[0].nodeValue + '\n';  
            } catch (e) {  
                output += people[i].childNodes[0].childNodes[0].nodeValue + '\t';  
                output += people[i].childNodes[1].childNodes[0].nodeValue + '\t';  
                output += people[i].childNodes[2].childNodes[0].nodeValue + '\n';  
            }  
        }  
        alert(output);  
    }  
}
```

그림 20-8 실행 결과



❖ XML 문서를 JSON으로 변경하는 플러그인 사용

- http://www.thomasfrank.se/xml_to_json.html
- XML 문서를 JSON 형식으로 변경하는 플러그인 다운

그림 20-9 XML 문서를 JSON 형식으로 변경하는 플러그인 사이트



❖ XML 문서를 JSON으로 변경하는 플러그인 사용 (2)

- Scripts 폴더에 파일 추가
- index.cshtml 파일의 head 태그 위치에 script 추가

그림 20-10 xml2json.js 파일을 scripts 폴더로 이동



코드 20-18 index.cshtml

```
<head>
  <title>Index</title>
  <script src="../../../Scripts/xml2json.js"></script>
  <script>
    /* 생략 */
  </script>
</head>
```



❖ xml2json 플러그인의 xml2json.parser() 메서드

- XML 문서 JSON 객체로 변환

```
{
  "people":{
    "person":[
      {
        "name":"윤인성",
        "gender":"남자",
        "part":"기타"
      },
      {
        "name":"연하진",
        "gender":"여자",
        "part":"기타"
      }
    ]
  }
}
```



❖ xml2json 플러그인의 xml2json.parser() 메서드

■ 활용 예제 코드

코드 20-19 index.cshtml: JSON 데이터 사용

```
if (request.status == 200) {  
    // XML을 JSON으로 변경합니다.  
    var json = xml2json.parser(request.responseText);  
  
    // 이용합니다.  
    for (var i in json['people']['person']) {  
        var name = json['people']['person'][i].name;  
        var gender = json['people']['person'][i].gender;  
        var part = part['people']['person'][i].part;  
    }  
}
```





Thank You !

모던 웹을 위한 Javascript jQuery 입문