

14장. Ajax



Ajax

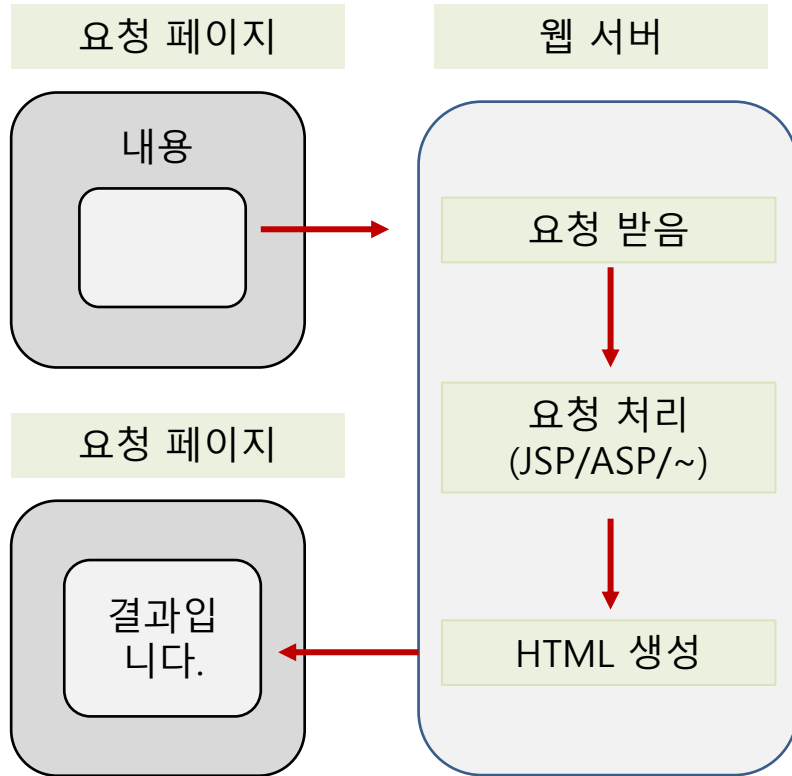


Ajax란?

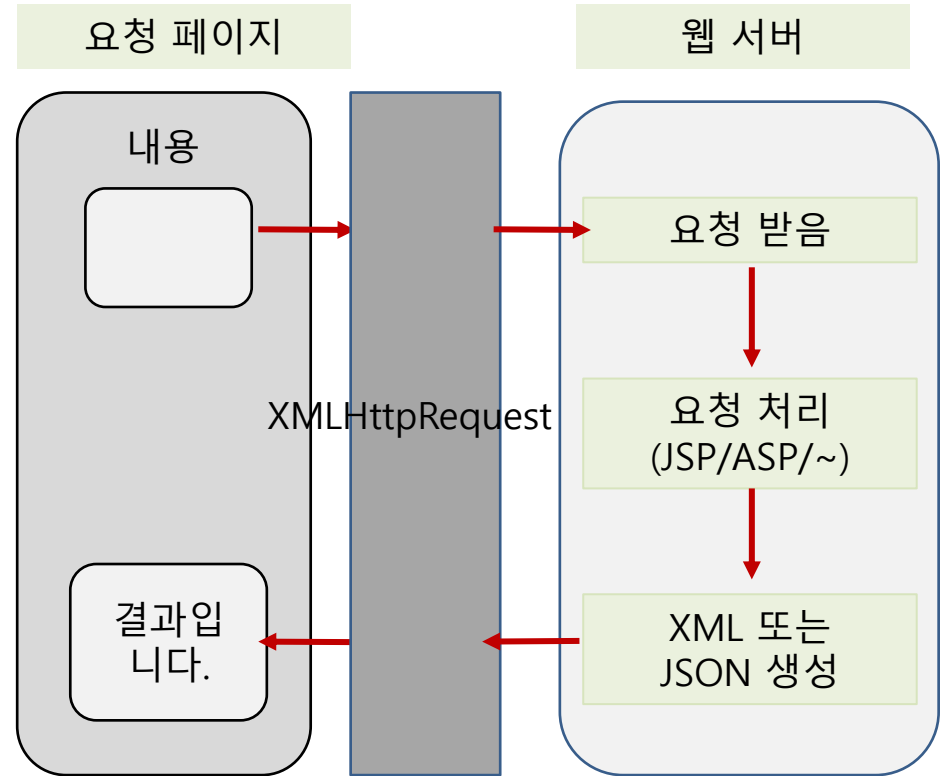
- **Ajax**란 Asynchronous Javascript(비동기 자바스크립트) + XML의 의미로 자바스크립트를 사용한 비동기 통신, 즉 클라이언트와 서버 간의 XML이나 JSON 데이터를 주고 받는 기술을 말한다.
- Ajax는 페이지 이동 없이 데이터 처리가 가능하며, 서버의 처리를 기다리지 않고 비동기 요청이 가능하다는 특징이 있다.

Ajax 개요

■ 웹 페이지 동작 방식 비교



전통적인 web 방식

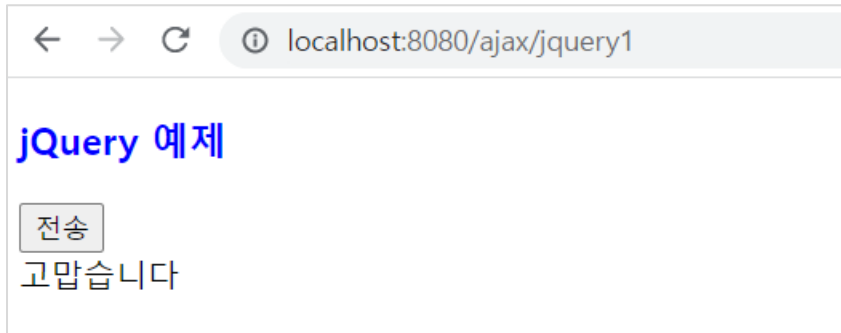


Ajax 처리 방식

jQuery(제이쿼리) Ajax 사용법.

```
$.ajax({  
  type: "post 또는 get",  
  dataType: " 서버에서 전송받을 데이터형식",  
  async: "true 또는 false",  
  url: "요청할 URL",  
  data: {서버로 전송할 데이터},  
  success: function(){  
    // 정상 요청, 응답 시 처리  
  },  
  error: function(){  
    // 오류 발생 시 처리  
  },  
});
```

❖ jQuery(제이쿼리) 예제



Controller 설계

```
@RequestMapping("/ajax/*")
@Controller
public class AjaxController {

    @GetMapping("/jquery1")
    public void jquery1() {
    }
```

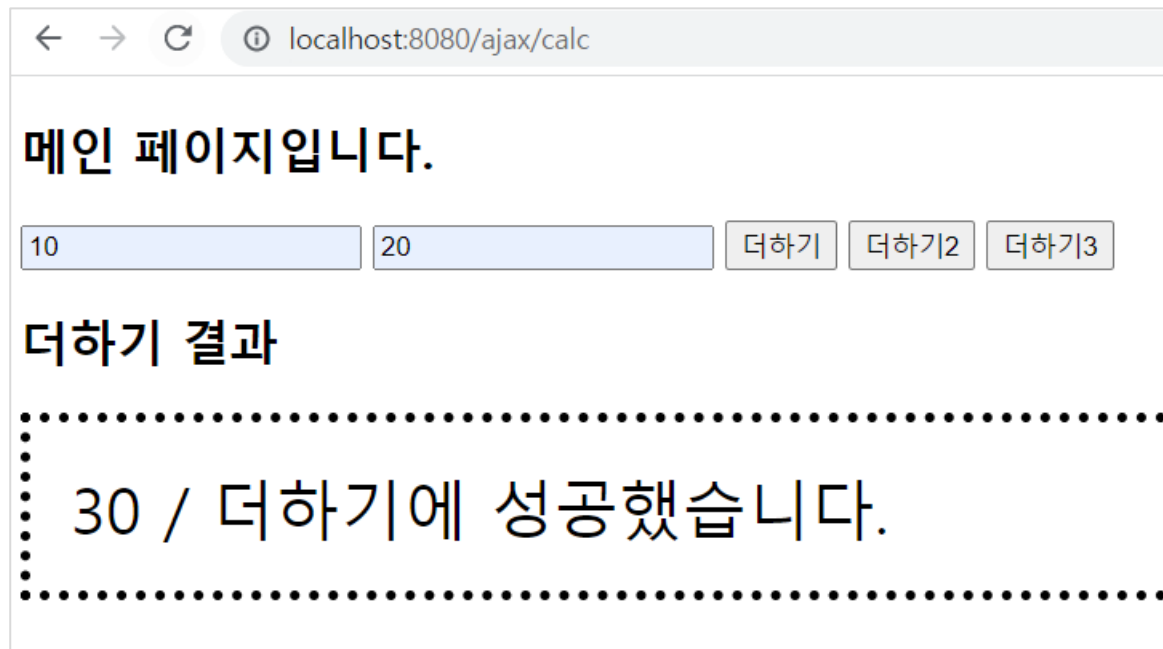
jQuery 실습

- ❖ jquery1.jsp – 제이쿼리 라이브러리 임포트함

```
<script src="https://code.jquery.com/jquery-3.6.1.js" integrity="sha256-  
crossorigin="anonymous"></script>  
<script>  
    $(document).ready(function(){  
        $("button").click(function(){  
            $("h3").css("color", "blue");  
            //$("#msg").empty().append("<p>고맙습니다</p>");  
            $("#msg").text("고맙습니다");  
        })  
    });  
</script>  
</head>  
<body>  
    <h3>jQuery 예제</h3>  
    <button type="button">전송</button>  
    <div id="msg"></div>  
</body>
```

ajax 실습

❖ 더하기 계산 프로그램



A screenshot of a web browser window with the address bar showing 'localhost:8080/ajax/calc'. The page content includes the text '메인 페이지입니다.' followed by two input fields containing '10' and '20', and three buttons labeled '더하기', '더하기2', and '더하기3'. Below these is the text '더하기 결과' and a dashed rectangular box containing the message '30 / 더하기에 성공했습니다.'

❖ Controller 설계

```
@RequestMapping("/ajax/*")
@Controller
public class AjaxController {

    @GetMapping("/jquery1")
    public void jquery1() {
    }

    @GetMapping("/calc")
    public String calcView() {
        return "/ajax/calc";
    }

    //더하기 계산
    @RequestMapping("/doPlus")
    @ResponseBody //데이터 출력
    public int doPlus(int num1, int num2) {
        return num1 + num2;
    }
}
```


ajax 실습

❖ /ajax/calc.jsp

```
<h2>메인 페이지입니다.</h2>
<form action="/ajax/doPlus" method="get" name="form1">
  <input type="text" name="num1" placeholder="숫자1">
  <input type="text" name="num2" placeholder="숫자2">
  <input type="submit" value="더하기">
  <input type="button" value="더하기2" onclick="calc()">
  <input type="button" value="더하기3" onclick="calc2()">
</form>

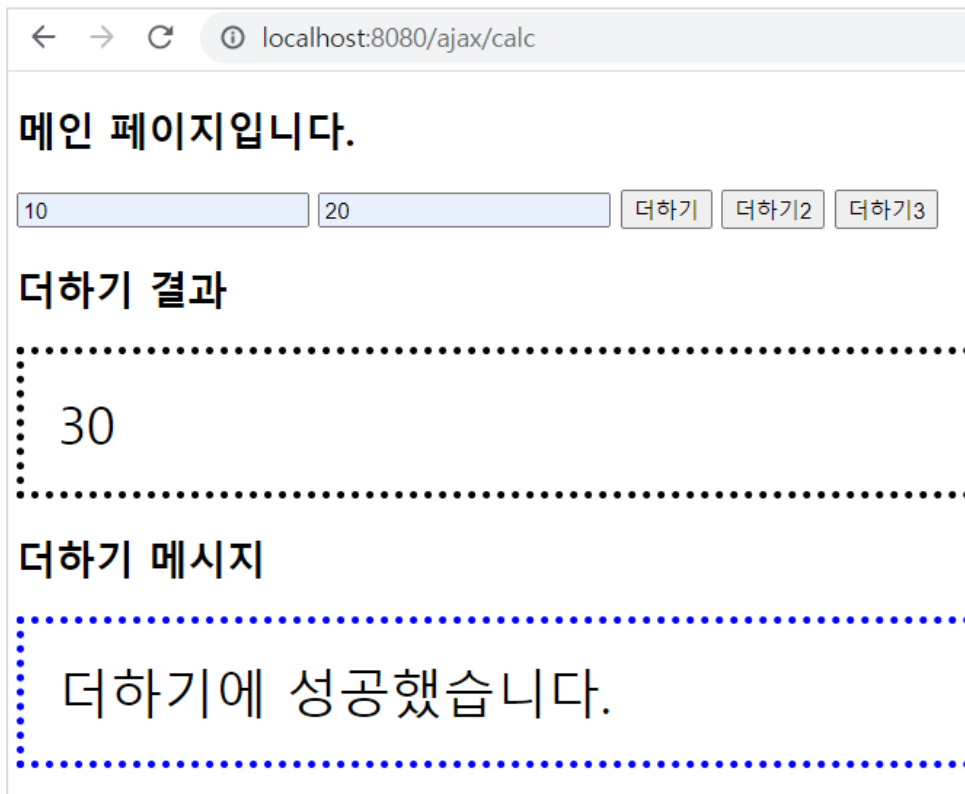
<h2>더하기 결과</h2>
<div class="result"></div>

<h2>더하기 메시지</h2>
<div class="result_msg"></div>
```

```
▼ WEB-INF
  classes
  > spring
  ▼ views
    ▼ ajax
      calc.jsp
      jquery1.jsp
      plus.jsp
    > board
    > user
```



❖ 더하기 결과 메시지 출력



A screenshot of a web browser window at the URL `localhost:8080/ajax/calc`. The page content is as follows:

- 메인 페이지입니다.
- Two input fields containing the numbers 10 and 20, followed by three buttons labeled "더하기", "더하기2", and "더하기3".
- The text "더하기 결과" (Addition Result) is displayed above a dotted rectangular box containing the number 30.
- The text "더하기 메시지" (Addition Message) is displayed above a dotted rectangular box containing the message "더하기에 성공했습니다." (Successful addition).

❖ Controller 설계

```
//더하기 및 메시지 처리
@RequestMapping(value="/doPlus", produces="application/text; charset=UTF-8")
@ResponseBody //데이터 출력
public String doPlus(int num1, int num2) {
    int sum = num1 + num2;
    String msg = "더하기에 성공했습니다.";
    return sum + " / " + msg;
}
```

❖ /ajax/calc.jsp

```
function calc2(){
    let form = document.form1;
    let num1 = form.num1.value;
    let num2 = form.num2.value;
    let action = form.action;

    $.ajax({
        type: "get",
        url: action,
        dataType: "text",
        data: {
            num1: num1, //데이터 전송
            num2: num2
        },
        success: function(data){
            //반환받은 값과 메시지 분리(배열)
            data = data.split("/");
            let sum = data[0];
            let msg = data[1];
            $(".result").text(sum);
            $(".result_msg").text(msg);
        },
        error: function(data){
            alert("error");
        }
    });
};
```



@ResponseBody

클라이언트에서 서버로 통신하는 메시지를 요청(request) 메시지라고 하며, 서버에서 클라이언트로 통신하는 메시지를 응답(response) 메시지라고 한다. 웹에서 화면전환(새로고침) 없이 이루어지는 동작들은 대부분 비동기 통신으로 이루어진다. 비동기통신을 하기위해서는 클라이언트에서 서버로 요청 메시지를 보낼 때, 본문에 데이터를 담아서 보내야 하고, 서버에서 클라이언트로 응답을 보낼 때에도 본문에 데이터를 담아서 보내야 한다. 이 본문이 바로 body 이다.

이때 본문에 담기는 데이터 형식은 여러가지 형태가 있는데 가장 대표적으로 사용되는 것이 JSON 이다.

ID 중복 검사

회원 가입시 ID 중복 검사

아이디	<input type="text" value="cloud"/> ID 중복
이미 가입된 ID입니다.	
비밀번호	<input type="text" value="PASSWORD"/>
비밀번호 확인	<input type="text" value="PASSWORD_CONFIRM"/>
이름	<input type="text"/>
성별	<input type="text" value="남"/> ▼
<input type="button" value="등록"/> <input type="button" value="취소"/>	

아이디	<input type="text" value="aabbcc"/> ID 중복
사용 가능한 ID입니다.	
비밀번호	<input type="text" value="PASSWORD"/>
비밀번호 확인	<input type="text" value="PASSWORD_CONFIRM"/>
이름	<input type="text"/>
성별	<input type="text" value="남"/> ▼
<input type="button" value="등록"/> <input type="button" value="취소"/>	

ID 중복 검사

- Mapper 설계

```
public interface MemberMapper {  
    public int checkID(String userid); //ID 중복 검사  
}
```

```
<!-- 아이디 중복 체크 -->  
<select id="checkID" resultType="int">  
    SELECT COUNT(*) FROM tbl_member  
    WHERE userid = #{userid}  
</select>
```

ID 중복 검사

- Service 설계

```
public interface MemberService {  
    public int checkID(String userid); //ID 중복 검사  
}
```

```
public class MemberServiceImpl implements MemberService {  
    @Override  
    public int checkID(String userid) {  
        return mapper.checkID(userid);  
    }  
}
```


ID 중복 검사

- Controller 설계

```
//ID 중복 체크
@GetMapping("/checkID")
@ResponseBody //데이터 전송 어노테이션
public int checkID(String userid) {
    int result = service.checkID(userid);
    return result;
}
```

- ID 중복 버튼 만들기 – signup.jsp

```
<tr>
    <td>아이디</td>
    <td>
        <input type="text" id="userid" name="userid" placeholder="ID">
        <input type="button" value="ID 중복" onclick="checkID()">
        <p id="check"></p>
    </td>
</tr>
```



ID 중복 검사

```
function checkID(){
    $.ajax({
        type: "get",
        url: "http://localhost:8080/member/checkID",
        dataType: "json",
        data: {"userid": $("#userid").val()}, //서버로 userid 보냄
        success: function(data){ //서버에서 응답 받음
            //console.log(data);
            if(data == 1){
                $("#check").text("이미 가입된 ID입니다.");
                $("#check").css({"color": "red", "padding-top": "5px"});
            }else if(data == 0){
                $("#idCheck").attr("value", "Y"); //버튼의 value를 "Y"로 변경
                $("#check").text("사용 가능한 ID입니다.");
                $("#check").css({"padding-top": "5px"});
            }
        },
        error: function(data){
            alert("에러 발생!!");
        }
    });
}
```



ID 중복 검사

- json 데이터 처리를 위한 라이브러리 추가

```
<dependency>  
  <groupId>com.fasterxml.jackson.core</groupId>  
  <artifactId>jackson-databind</artifactId>  
  <version>2.10.3</version>  
</dependency>
```

유효성 검사

- 필드 항목에 id 선택자 사용하기 – signup.jsp

```
<form action="/member/signup" method="post" id="regForm" name="regForm"
  onsubmit="return checkForm()">
  <table class="tbl_signup">
    <tr>
      <td>아이디</td>
      <td>
        <input type="text" id="userid" name="userid" placeholder="ID">
        <button type="button" id="idCheck" value="N" onclick="checkID()">ID 중복</button>
        <p id="check"></p>
      </td>
    </tr>
    <tr>
      <td>비밀번호</td>
      <td><input type="password" id="userpw" name="userpw" placeholder="PASSWORD"></td>
    </tr>
    <tr>
      <td>비밀번호 확인</td>
      <td><input type="password" id="userpw_confirm" name="userpw_confirm" placeholder="PA.
    </tr>
    <tr>
      <td>이름</td>
      <td><input type="text" id="username" name="username"></td>
    </tr>
```

유효성 검사

- 유효성 검사 – validation.js

```
function checkForm(){
    //alert("test");
    let id = document.getElementById("userid");
    let pwd1 = document.getElementById("userpw");
    let pwd2 = document.getElementById("userpw_confirm");
    let name = document.getElementById("username");
    let idChkVal = document.getElementById("idCheck");

    //정규식 변수 할당
    let regExpId = /^[a-zA-Z0-9]*$/ //영문자, 숫자만(^-시작, *-반복)
    let regExpPwd1 = /[a-zA-Z0-9]/ //영문자, 숫자
    let regExpPwd2 = /[~!@#$$%^&*()_+]/ //특수문자
    let regExpPwd3 = /[ㄱ-ㅎㅏ-ㅣ가-힣]/ //한글

    if(id.value.length < 4 || id.value.length > 12 || !regExpId.test(id.value)){
        alert("아이디는 영문자, 숫자 포함 4-12자 이하로 입력해주세요 ");
        userid.focus();
        userid.select();
        return false;
    }
}
```

유효성 검사

- 유효성 검사 – validation.js

```
if(pwd1.value.length < 8 || pwd1.value.length > 12 || !regExpPwd1.test(pwd1.value)
    || !regExpPwd2.test(pwd1.value) || regExpPwd3.test(pwd1.value)){
    alert("비밀번호는 영문자, 숫자, 특수문자 포함 8-12자 이하로 입력해주세요 ");
    userpw.focus();
    userpw.select();
    return false;
}
if(pwd1.value != pwd2.value){
    alert("비밀번호를 동일하게 입력해주세요");
    userpw_confirm.select();
    return false;
}
if(name.value == ""){
    alert("이름을 입력해주세요");
    username.focus();
    return false;
}
if(idChkVal.value == "N"){
    alert("아이디 중복 확인을 해주세요");
    return false;
}
```