Ajax 시작하기

Ajax 란?

장점

- **비 동기식 방식으로 웹서버의 응답을 기다리지 않고 데이터를 빠르게 처리하는 개발 기법**
- 기존 방식은 웹서버의 응답으로 페이지의 깜빡임(새로고침)이 있고 난 뒤 데이터가 처리 됐지만 Ajax 는 페이지 리로딩 없이 데이터를 처리
- 대표적 기능으로 네이버나 구글의 실시간 검색이나 검색어 자동완성 등이 있습니다.

단점

- 페이지의 리로딩이 없기 때문에 한 페이지에서 지속적으로 사용시 리소스가 계속 쌓여 페이지가 느려짐
- 스크립트로 되어 있기 때문에 에러 발생 시 디버깅에 어려움이 있습니다.

AJAX 란 서버로부터 데이터를 가져와 전체 페이지를 새로 고치지 않고 일부만 로드할 수 있게 하는 기법이다. 본래 비동기 요청을 보내는 데 필요한 기술, 즉 **Asynchronous JavaScript AND XML** 의 약자였으나 이후 브라우저 내에서 비동기 기능을 제공하는 모든 기법을 통칭하게 되었다. 간단히 말하면, 서버측 Scripts 와 통신하기 위한 XMLHttpRequest 객체를 사용하는 것을 말합니다. 서버측으로 다양한 형식(JSON, XML, HTML 및 일반 텍스트 형식 등)의 정보를 주고 받을 수 있습니다.

AJAX 의 강력한 특징은 페이지 전체를 리프레쉬(새로고침) 하지 않고서도 수행 되는 "비동기성"입니다. 이러한 비동기성을 통해 사용자의 Event 가 있으면 전체 페이지가 아닌 일부분만을 업데이트 할 수 있게 해줍니다.

다시 말해 아래와 같이 두 가지로 정리됩니다.

- 페이지 일부분을 업데이트 하기 위한 정보를 서버에 요청할 수 있다.
- 서버로부터 받은 데이터로 작업을 한다.

동기처리모델(synchronouse processing model)의 개념

동기처리모델에서 브라우저는 <script> 태그를 만나면 스크립트를 로드하고 처리하기 전까지 다른 작업을 중단한다. 브라우저는 급하게 다른 작업으로 바로 넘어가지 않고, <script> 태그의 내용이 로드되고 처리될 때까지 기다려준다. 즉, 브라우저는 스크립트가 서버로부터 데이터를 수집하고 이를 처리한 후 페이지의 나머지 부분이 모두 로드될 때까지 대기한다.

비동기처리모델(asynchronous processing model)의 개념

Ajax는 비동기처리모델을 사용한다. 페이지가 로드되는 동안 Ajax를 사용하면, 먼저 브라우저는 서버에 데이터를 요청한다. 그리고 일단 데이터를 요청했다면, 페이지의 나머지를 계속해서 로드하고 페이지와 사용자의 상호작용을 처리한다. 이런 방식을 비동기 처리모델 혹은 넌블로킹 모

델(non-blocking model)이라고 한다. 브라우저는 페이지를 표시할 서드파티 데이터를 기다리지 않는다. 서버가 데이터를 전달하면 이벤트가 발생한다. 그러면 이벤트는 데이터를 처리할 함수를 호출한다.

Ajax 활용 예시

라이브검색 : 오늘날 검색사이트의 대부분이 사용하는 기술로 검색어를 입력하는 동시에 자동 단어완성기능으로 검색결과가 바로 나타나도록하는 기능이다.

사용자 정보 표시 : 사용자가 생성한 콘텐츠(최근 트윗이나 사진 등)를 다른 웹사이트의 독자들에게 노출하면 각자 자신들의 서버에 데이터를 수집할 수 있는 기능을 제공한다. 또는 회원가입시 중복아이디 체크 기능이나 온라인쇼핑몰에서 장바구니에 원하는 상품을 추가했을 때 페이지 이동이나 전체 페이지에 대한 새로고침 없이 물품정보만 추가되는 기능을 구현하고자 할 때도 활용한다.

1 단계 – HTTP request 만들기

JavaScript 를 이용하여 서버로 보내는 HTTP request 를 만들기 위해서는 이런 기능을 제공하는 클래스 인스턴스가 필요하다. 이런 클래스는 Internet Explorer 에서는 XMLHTTP 라고 불리는 ActivX object 를 말한다. 그러면 Mozzlia, Safari 나 다른 브라우저는 Microsoft 의 ActiveX 객체의 method 와 property 를 지원하기 위해 XMLHttpRequest 클래스를 구현 하고 있습니다.
따라서, 다음과 같이 하면 모든 브라우저에서 사용할 수 있는 인스턴스를 만들 수 있습니다.

```
var httpRequest;
if (window.XMLHttpRequest) { // 모질라, 사파리등 그외 브라우저, ...
    httpRequest = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE 8 이상
    httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
}
```

다음으로는, 서버로 보낸 요청(request)에 대한 응답을 받았을 때 어떤 동작을 할 것인지에 대한 사항을 정해야 합니다. 이 단계에서는 단순하게 위에 생성된
httpRequest 의 onreadystatechange property 를 통해 어떤 함수가 그 동작을 수행 할 것인지 아래와 같이 할당하면 됩니다.

```
httpRequest.onreadystatechange = nameOfTheFunction;
```

주목할 사항으로는 위에서는 해당 함수를 수행하는 것이 아니라 단순하게 어떤 함수가 불릴 것인지만 지정한다는 점입니다. 단순하게 그 함수를 지정하는 것이므로 그 함수로 어떠한 변수도

전달하지 않습니다. 또한 단순하게 함수를 연결하면 되기 때문에 아래와 같이 JavaScript 에서 사용되는 "임의 함수(anonymous functions)"방법으로 직접적인 함수 본체를 기입해도 됩니다.

```
httpRequest.onreadystatechange = function(){
    // process the server response
};
```

다음으로, 위와 같이 서버로 부터 응답을 받은 후의 동작을 결정 한 후에는 실질적으로 요청(request)를 하는 것 입니다. 요청(request)을 하기 위해서는 HTTP request class 의 open()과 send()를 아래와 같이 호출하여야 합니다.

```
httpRequest.open('GET', 'http://www.example.org/some.file', true);
httpRequest.send(null);
```

- **open() 메소드의 첫번째 파라미터**는 HTTP 요구 방식(request method) - GET, POST, HEAD 중의 하나이거나 당신의 서버에서 지원하는 다른 방식 입니다. 이 파라미터는 HTTP 표준에 따라 **모두 대문자로 표기**하십시오. 그렇지 않으면 (파이어폭스와 같은) 특정 브라우저는 이 요구를 처리하지 않을 수도 있습니다. 가능한 HTTP 요구 방식에 대한 정보는 W3C 명세를 참고하기 바랍니다.
- **두번째 파라미터**는 요구할 페이지의 URL 입니다. 보안을 위해 서드 파티 도메인 상의 페이지를 호출할 수는 없습니다. 요구하는 모든 페이지에 정확한 도메인 네임을 사용하십시오. 그렇지 않으면 open() 메소드를 호출할 때 'permission denied' 에러가 발생할 수 있습니다. 일반적인 오류는 당신의 사이트에 domain.tld 와 같은 형태로 접근하는 것 입니다. 이러한 경우 www.domain.tld 와 같은 형태로 페이지를 요구하기 바랍니다.
- **세번째 파라미터**는 요구가 비동기식(Asynchronous)으로 수행될 지를 결정합니다. 만약 이 파라미터가 TRUE 로 설정된 경우에는 자바스크립트 함수의 수행은 서버로부터 응답을 받기 전에도 계속 진행됩니다.

send() 메소드의 파라미터는 POST 방식으로 요구한 경우 서버로 보내고 싶은 어떠한 데이터라도 가능합니다. 데이터는 서버에서 쉽게 parse 할 수 있는 형식(format)이어야 합니다.
예를 들자면 아래와 같습니다.

```
"name=value&anothername="+encodeURIComponent(myVar)+"&so=on"
```

또는 JSON, SOAP 등과 같은 다른 형식으로도 가능합니다.
만약 "POST" type 을 보내려 한다면, 요청(request)에 MINE type 을 설정 해야 합니다.
예를 들자면 send()를 호출 하기 전에 아래와 같은 형태로 send()로 보낼 쿼리를 이용해야 합니다.

```
http_request.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
```

2 단계 – 서버 응답에 대한 처리

요구(request)를 보냈을 때 이미 응답을 처리하기 위한 자바스크립트 함수의 이름을 지정했었다는 것을 기억하기 바랍니다.

http_request.onreadystatechange = nameOfTheFunction;

이제 이 함수가 어떤 일을 하는지 보기로 합시다. 먼저, 해당 함수에서는 요구의 상태값을 검사할 필요가 있습니다. 만약 상태값이 4 라면, 서버로부터 모든 응답을 받았으며 이를 처리할 준비가 되었다는 것을 뜻합니다.

```
if (http_request.readyState == 4) {
    // 이상 없음, 응답 받았음
} else {
    // 아직 준비되지 않음
}
```

readyState 가 가질 수 있는 모든 값의 목록은 아래와 같습니다:

- 0 (uninitialized)
- 1 (loading)
- 2 (loaded)
- 3 (interactive)
- 4 (complete)

다음으로 검사할 것은 HTTP 서버 응답의 상태 코드입니다. 가능한 모든 코드 값의 목록은 W3C 사이트에서 확인할 수 있습니다. 여기서는 단지 정상적으로 처리된 상태를 나타내는 200 의 경우만을 검사합니다.

```
if (http_request.status == 200) {
    // 이상 없음!
} else {
    // 요구를 처리하는 과정에서 문제가 발생되었음
    // 예를 들어 응답 상태 코드는 404 (Not Found) 이거나
    // 혹은 500 (Internal Server Error) 이 될 수 있음
}
```

이제 요구와 그에 대한 응답에 대한 상태 코드를 검사했으므로, 서버에서 받은 데이터를 통해 원하는 작업을 수행할 수 있다. 그리고 위의 응답 데이터에 접근하기 위한 2 가지 옵션이 아래와 같이 있습니다.

- **http_request.responseText** – 서버의 응답을 텍스트 문자열로 반환할 것이다.

- **http_request.responseXML** – 서버의 응답을 XMLDocument 객체로 반환하며 당신은 자바스크립트의 DOM 함수들을 통해 이 객체를 다룰 수 있을 것이다.

위의 단계는 비동기식 요구(asynchronous request)를 사용했을 경우에 대한 설명입니다. (즉, send()의 세 번째 변수가 "true"일경우). 동기식(Synchronous) 요구(request)를 사용한다면 함수를 명시할 필요 없이 send()를 호출에 의해 반환되는 data 를 바로 사용 할 수 있습니다. 이는 스크립트가 send()를 호출할 때 멈춰지며 서버의 응답이 완료될 때까지 기다립니다.

3 단계 – 간단한 예제

이제 이들을 한데 모아서 간단한 HTTP 요구를 수행해 보겠습니다. 우리가 작성할 자바스크립트는 "I'm a test." 라는 문장이 적힌 test.html 이라는 HTML 문서를 요구해서 문서의 내용을 파라미터로 alert() 함수를 호출할 것입니다.

```
<span id="ajaxButton" style="cursor: pointer; text-decoration: underline">
  Make a request
</span>
<script type="text/javascript">
(function() {
  var httpRequest;
  document.getElementById("ajaxButton").onclick = function() { makeRequest('test.html'); };

  function makeRequest(url) {
    if (window.XMLHttpRequest) { // Mozilla, Safari, ...
      httpRequest = new XMLHttpRequest();
    } else if (window.ActiveXObject) { // IE
      try {
        httpRequest = new ActiveXObject("Msxml2.XMLHTTP");
      }
      catch (e) {
        try {
          httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
        }
        catch (e) {}
      }
    }

    if (!httpRequest) {
      alert('Giving up :( Cannot create an XMLHTTP instance');
```

```
        return false;
      }
      httpRequest.onreadystatechange = alertContents;
      httpRequest.open('GET', url);
      httpRequest.send();
    }

    function alertContents() {
      if (httpRequest.readyState === 4) {
        if (httpRequest.status === 200) {
          alert(httpRequest.responseText);
        } else {
          alert('There was a problem with the request.');
        }
      }
    }
})();
</script>
```

이 예제에서:

- 사용자는 브라우저 상의 "Make a request" 라는 링크를 클릭합니다;
- 그러면 같은 디렉토리 내의 HTML 파일의 이름인 test.html 를 파라미터로 하여 makeRequest() 함수를 호출합니다;
- 브라우저는 서버로 요구를 보내고 onreadystatechange 에 설정된 alertContents() 함수가 수행됩니다;
- alertContents() 함수는 서버로부터 응답을 받았는지와 정상적으로 처리된 응답인지를 검사하여 그 경우 test.html 파일의 내용을 파라미터로 alert() 함수를 호출합니다.

주 1 : Internet Explorer 에서 동작하는 page 를 만들 경우에는 **Mozilla 용**에서 추가적으로 다음과 같은 작업이 필요합니다. 정적 XML 파일이 아니라, XML 을 받기 위한 request 를 보낼 때, 몇가지 응답 헤더를 설정해야 합니다. **헤더에 "Content-Type: application/xml"** 설정을 하지 않을 경우, IE 는 접근하고자 하는 XML 요소 다음 라인을 수행한 직후 "Object Expected" JavaScript error 를 발생시킬 것입니다.

주 2 : **헤더에 "Cache-Control: no-cache"**를 설정하지 않는다면, 브라우저는 debugging "challenging."으로 만들고 해당 응답을 유지하며, 다시는 요청을 받지 않을 것입니다. 물론 항상 달라지는 "GET"의 추가 인자(시간 정보나, 랜덤 넘버)를 추가하여 cache 되지 않도록 작업을 할 수도 있습니다.(bypassing the cache 를 참고하세요)

주 3 : 만약 http_request 변수가 전역적으로 사용되면, makeRequest() 함수를 호출하는 여러 함수들 사이에서 경쟁 상태가 발생할 수 있으며, 이 경우 다른 데이터를 덮어쓰게 됩니다. 이를 방지하기 위해서는 **http_request 변수를 함수 내의 지역 변수로 선언하여 alertContent() 함수에 넘겨야 합니다**.

주 4 : 브라우저가 멈춰버리는 통신 에러 이벤트에서, "status" field 를 접근하려 할 때는 onreadystatechange 메소드에서 예외에러를 발생시킬 것입니다. **Try...catch 구문 안의 If..then statement 를 잘 감싸져 있는지 확인**하세요.

```
function alertContents(httpRequest) {
  try {
    if (httpRequest.readyState === 4) {
      if (httpRequest.status === 200) {
        alert(httpRequest.responseText);
      } else {
        alert('There was a problem with the request.');
      }
    }
  }
  catch( e ) {
    alert('Caught Exception: ' + e.description);
  }
}
```

4 단계 – "The X-Files" or Working with the XML Response

앞의 예제에서 HTTP 요구(request)에 대한 응답을 받은 후에 리퀘스트 오브젝트(request object) 중 reponseText 프로퍼티를 사용했습니다. 그리고 reponseText 은 test.html 파일의 내용을 가지고 있습니다. 이제 **responseXML** 를 사용해 봅시다.

첫째로, 나중에 서버에 요구할 수 있는 유효한 XML 문서를 만들어 봅시다. 이 문서(test.xml)은 아래와 같은 내용을 담고 있습니다.

```
<?xml version="1.0" ?>
<root>
    I'm a test.
</root>
```

실행 스크립트에서 서버에 대해 요청하는 라인을 아래와 같이 바꿔줘야 합니다.

```
...
onclick="makeRequest('test.xml')">
...
```

그 다음에 alertContents()함수에서, alert()함수를 실행하는 라인,
즉, alert(http_request.responseText); 을 아래와 같이 바꿔줘야 합니다.

```
var xmldoc = http_request.responseXML;
var root_node = xmldoc.getElementsByTagName('root').item(0);
alert(root_node.firstChild.data);
```

이 방법은 responseXML 에 의한 XMLDocument 오브젝트를 가져왔습니다. 그리고 XML 문서에 포함된 데이터를 가져오기 위해서 DOM methods 를 사용했습니다.

DOM methods 에 대한 더 자세한 사항은 Mozilla's DOM implementation 문서를 확인하십시오

예제

우선 Ajax 를 사용하기 위해서는 jquery 가 필요합니다. 다음과 같이 jquery 에서 직접 제공하는 주소를 사용하거나 js 파일을 구하셔서 import 해야 합니다.

```
<script src="http://code.jquery.com/jquery-1.7.js" type="text/javascript"></script>
```

HTML 코드 :

```
<html>
<head>
	<title>Home</title>
	<script src="http://code.jquery.com/jquery-1.7.js" type="text/javascript"></script>
	<script type="text/javascript">
	function AjaxEx(){
		var name = "name="+document.check.name.value;
		$.ajax({
			url : "/check",
			type : "post",
			data : name,
			dataType : "json",
			success : function(data) {
				document.getElementById("result").innerHTML = data+"님
반갑습니다.";
				},
			error : function(request) {
				alert("실패");
			}
		});
		}
	</script>
</head>
<body>
<h3>Ajax 예제 입니다..</h3>
<form action="" name="check" method="post">
<input type="text" id="name">
<input type="button" value="클릭" onclick="AjaxEx()">
<div id="result"></div>
</form>
</body>
```

```
</html>
```

JAVA 코드 :

```
@RequestMapping(value = "/check", method = RequestMethod.POST)
public void check(HttpServletRequest request, HttpServletResponse resp) throws Exception {
                        String name = request.getParameter("name");
                        JSONObject obj = new JSONObject();
                        JSONArray ja = new JSONArray();
                        ja.add(name);
                        PrintWriter out = resp.getWriter();
                        out.print(ja.toString());
            }
```

스프링 기본 프로젝트를 생성한 후 실행 시키면 나오는 Home 화면에 다음과 같이 JSP 단을 작성해 줍니다.
그리고 텍스트박스에 이름을 넣고 클릭 버튼을 누르면 스크립트 함수를 통해 Ajax 를 수행하고 결과를 리턴해 줍니다.

DB 값 json 으로 가지고 와서 ajax 로 데이터 뿌리기

이제 json 데이터를 가지고 와서 ajax로 출력하도록 해보자

먼저 servlet_context.xml 에 다음과 같이 설정되어 있는지 확인한다.

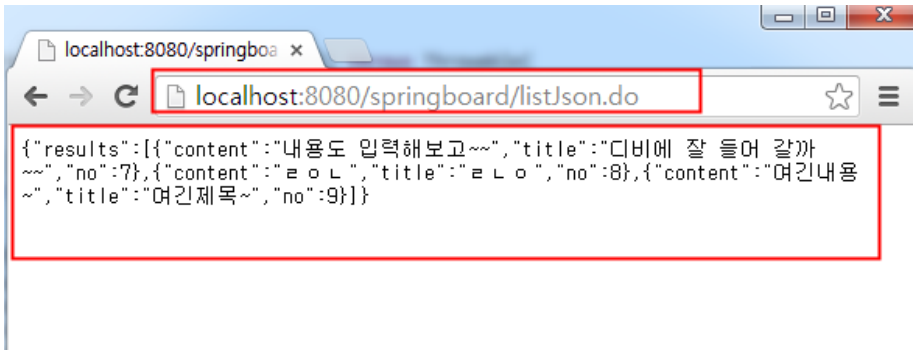
servlet_context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/s
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-co
        http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.1.xs

    <context:component-scan base-package="spring.board" use-default-filters="false">
        <context:include-filter type="annotation" expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

    <!-- Enables the Spring MVC @Controller programming model -->
    <mvc:annotation-driven>
        <mvc:message-converters register-defaults="true">
            <bean class = "org.springframework.http.converter.json.MappingJacksonHttpMessageConverter">
                <property name="supportedMediaTypes" value = "text/plain;charset=UTF-8" />
            </bean>
        </mvc:message-converters>
    </mvc:annotation-driven>

    <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views direct
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/jsp/" />
        <property name="suffix" value=".jsp" />
    </bean>

</beans>
```

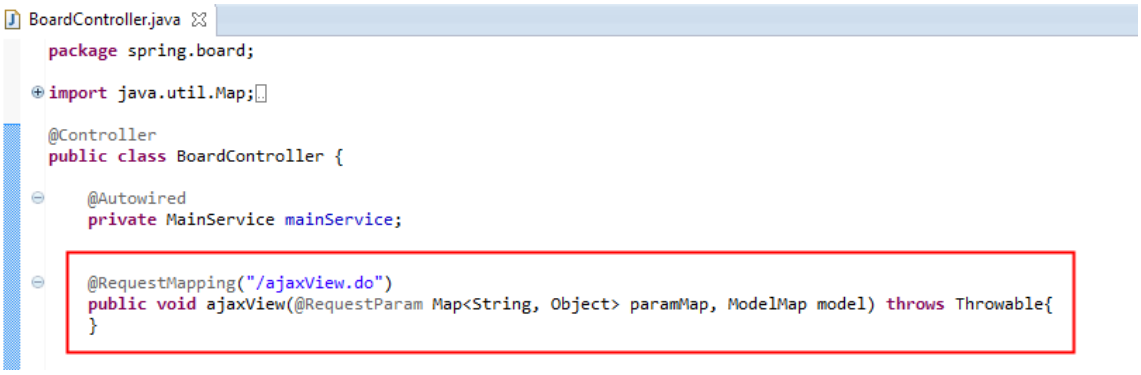
설정이 끝났으니 이제 Controller을 추가하자.
기존 list와 다른 점은 @ResponseBody 추가 되었고, return model; 이렇게 리턴 값이 있다는 것이다.
view 페이지는 만들지 않아도 된다.

```
@Controller
public class BoardController {

    @Autowired
    private MainService mainService;


    @RequestMapping("/list.do")
    public void list(@RequestParam Map<String, Object> paramMap, ModelMap model) throws Throwable{
        model.put("results", mainService.getList(paramMap));
    }

    @RequestMapping("/listJson.do")
    public  @ResponseBody Map<?,?> listJson(@RequestParam Map<String, Object> paramMap, ModelMap model) throws Throw
        model.put("results", mainService.getList(paramMap));
        return model;
    }
```

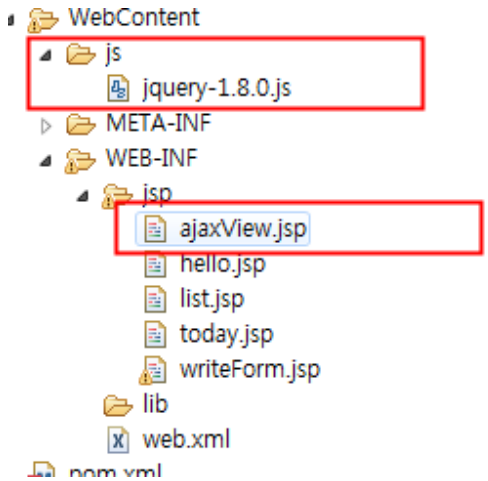
자 그럼 json 으로 나오는지 출력해보자



이제 ajax을 이용하여 리스트를 출력해보자.
테스트 할 Controller 추가 하자.



jquery을 사용하기 위헤 관련 js 파일과 view 페이지도 추가 한다.

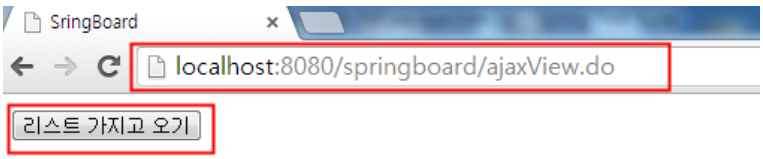


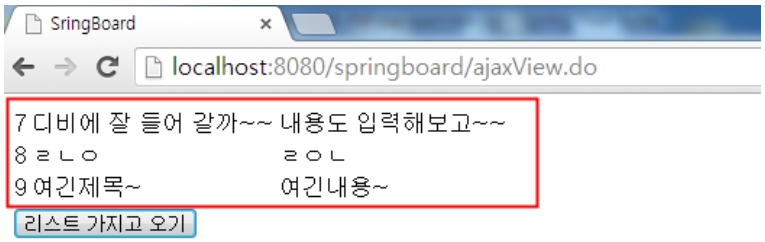
ajax을 호출하여 리스트를 출력하도록 코딩을 하고

ajaxView.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<html>
<head>
    <title> SringBoard</title>
    <script type="text/javascript" src="/springboard/js/jquery-1.8.0.js" charset="utf-8"></script>
    <script type="text/javascript">
        function viewAjaxList() {
            $.ajax({
                type     : "POST"
            ,   url      : "/springboard/listJson.do"
            ,   dataType : "json"
            ,   data     : ""
            ,   success  : function(result) {
                    $.each(result,function(key) {
                        var list = result[key];

                        var content = "<table>";

                        for( i = 0;i < list.length ;i++) {
                            content += "<tr>";
                            content +="<td>" + list[i].no + "</td>";
                            content +="<td>" + list[i].title + "</td>";
                            content +="<td>" + list[i].content + "</td>";
                            content +="</tr>";

                        }
                        content += "<table>";

                        $("#ajaxList").html(content);
                    });
                }
            });


        }

    </script>

</head>
<body>
<div id="ajaxList"></div>

<input type="button" value="리스트 가지고 오기" onclick="viewAjaxList() ">

</body>
</html>
```

실행 확인.




SringBoard
localhost:8080/springboard/ajaxView.do
7 디비에 잘 들어 갈까~~ 내용도 입력해보고~~
8 ㄹㄴㅇ ㄹㅇㄴ
9 여긴제목~ 여긴내용~
리스트 가지고 오기

[jQuery] ajax 데이터 넘기기 예제.

간단하게 input 값들을 문자열, 배열로 한꺼번에 넘기는 예제다.

HTML 코드

```
<input type="hidden" id="userId" value="abcd">
<input type="checkbox" name="hobby" value="독서">
<input type="checkbox" name="hobby" value="운동">
<input type="checkbox" name="hobby" value="전시관람">
```

javascript 단에서 데이터를 불러와서 ajax 로 넘긴다,,

```
function ajaxExample(){
    // 사용자 ID를 갖고 온다.
    var userId = $("#userId").val();

    // name이 같은 체크박스의 값들을 배열에 담는다.
    var checkboxValues = [];
    $("input[name='hobby']:checked").each(function(i) {
        checkboxValues.push($(this).val());
    });

    // 사용자 ID(문자열)와 체크박스 값들(배열)을 name/value 형태로 담는다.
    var allData = { "userId": userId, "checkArray": checkboxValues };

    $.ajax({
        url:"goUrl.do",
        type:'GET',
        data: allData,
        success:function(data){
            alert("완료!");
            window.opener.location.reload();
            self.close();
        },
        error:function(jqXHR, textStatus, errorThrown){
            alert("에러 발생~~ ₩n" + textStatus + " : " + errorThrown);
            self.close();
        }
    });
}
```

Spring Controller 단에서 데이터를 받을 땐, 배열은 배열로, 문자열은 문자열로 받으면 된다.

```
@RequestParam(value="checkArray[]") List<string> arrayParams, @RequestParam(value="userId")
String userId
```

만약 @RequestParam HashMap param 으로 받을 경우, 배열은 첫번째 값만 넘어간다. 그러니 배열은 반드시 List 로 받도록 한다.

JSONP 의 기본원리

JSONP 는 한 웹페이지에서 도메인이 다른 웹페이지로 데이터를 요청할 때 사용하는 자바스크립트 개발 방법론입니다.
기본적으로 웹 브라우저는 도메인이 다른 웹 페이지로는 Ajax 등의 방법으로 접근하지 못하게 제한하고 있는데, 이것을 **동일출처원칙**(Same-origin policy)이라고 합니다. 그러나 실무에서는 부득이 다른 도메인에 연결된 서버로 데이터를 요청해야만 하는 상황을 만나게 됩니다.
JSONP 는 바로 이러한 경우에 동일출처원칙을 회피하는 일종의 편법 입니다.

JSONP 는 동일출처원칙을 회피하기 위해 <script> 요소를 이용합니다.
본래 자바스크립트에서는 Ajax 를 비롯한 어떠한 방법으로도 직접 다른 도메인의 웹페이지로 데이터를 요청할 수 없습니다. 그러나 <script> 요소는 도메인이 다른 스크립트 파일이라 하더라도 임베드할 수 있기 때문에, 이 성질을 이용하는 것입니다.

jQuery 를 사용하여 간편하게 JSONP 를 구현한 방법은 Ajax 를 구현할 때와 유사한 방식으로 쉽고 간편하게 JSONP 를 구현할 수 있게 해주지만, 한편으로는 JSONP 의 기본원리에 대해서는 소홀하게 만드는 양면성을 가지고 있습니다.

JSONP 의 기본원리를 설명하기 위하여, 순수 자바스크립트로 JSONP 를 구현한 예제코드를 작성합니다. 또한 Ajax 와의 비교를 통해 JSONP 의 효용과 한계에 대해서 짚어보겠습니다. 그리고 실무에서 활용하기 좋은 jQuery 에서의 JSONP 구현방법까지 살펴봅니다.

예제코드 개요

회원명부에서 이름으로 검색하는 웹페이지를 작성하려고 합니다. 그런데 한 가지 문제가 있습니다. 메인 웹 서버에는 a.epiloum.net 도메인이 연결되어 있는데 반해, 회원명부 DB 는 b.epiloum.net 도메인에 연결된 서버에 있다는 것입니다. 동일출처원칙으로 인하여 일반적인 방법으로는 a.epiloum.net 의 웹페이지가 b.epiloum.net 에 있는 웹페이지를 호출할 수는 없습니다.
(이러한 상황을 타개하는 가장 간편한 방법은, a.epiloum.net 에서 b.epiloum.net 의 DB 를 접근할 수 있도록 접근권한을 조정하는 것입니다. 하지만 웹 서버와 DB 서버가 여러 대로 분산되어 있는 대형 웹 서비스라면, 이것 또한 녹록치 않을 것입니다.)
위와 같은 상황을 JSONP 를 통하여 극복해보도록 하겠습니다.

클라이언트측 예제코드 – http://a.epiloum.net/index.html

아래는 JSONP 로 데이터를 요청할 검색화면 웹페이지의 예제코드입니다. 이 소스코드 파일은 a.epiloum.net 라는 도메인 아래에 위치할 것입니다.

```
<!DOCTYPE html>
<html lang="ko">
```

```
<head>
<title>JSONP Sample Code</title>
<meta charset="utf-8">
<script type="text/javascript">
function find()
{
    var j = document.createElement('script');
    var s = encodeURIComponent(document.getElementById('str').value);
    var u = 'http://b.epiloum.net/find.php?callback=loadList&s=' + s;

    j.setAttribute('src', u);
    j.setAttribute('type', 'text/javascript');
    document.getElementsByTagName('body')[0].appendChild(j);

    return false;
}

function loadList(arr)
{
    var o = document.getElementById('list');
    var l;

    while(o.childNodes.length)
    {
        o.removeChild(o.lastChild);
    }

    for(var i=0; i<arr.length; i++)
    {
        l = document.createElement('li');
        l.appendChild(document.createTextNode(arr[i]));
        o.appendChild(l);
    }
}
</script>
</head>
<body>
<input id="str" type="text" size="50" />
<input id="btn" type="button" value="입력" onclick="return find()" />
```

```
<ul id="list"></ul>
</body>
</html>
```

실제로 코드를 웹 서버에 올려 접속해보면, 텍스트 입력란과 버튼이 표시된 화면을 볼 수 있습니다. 텍스트 입력란에 적당한 텍스트를 입력하고 버튼을 누르면, find()라는 자바스크립트 함수가 실행될 것입니다. 이 함수는 새로운 <script> 요소를 생성하여 <body> 요소 아래에 추가하는 역할을 합니다.
여기서 주목할 것은 <script>의 src 속성인데, URL 의 도메인이 현재 도메인과는 다른 b.epiloum.net 임을 확인할 수 있습니다. GET 파라메터 또한 확인해두는데, callback 이라는 파라메터에는 "loadList"라는 문자열이 들어가 있는데, 이것과 같은 이름을 가진 loadList()라는 함수가 있다는 점을 기억하기 바랍니다. 이어서 s 라는 파라메터에는 사용자가 텍스트 입력란에 타이핑한 값이 들어가게 됩니다.
마지막으로 위 예제코드 어디에서도 호출하는 부분이 없는 loadList() 함수가 있습니다. 이 함수는 인자로 받은 배열의 내용을 <ul id="list"> 요소 아래에 표시하는 역할을 합니다.

서버측 예제코드 – http://b.epiloum.net/find.php

이제 앞서의 예제코드에서 <script> 요소를 생성하여 호출하는 find.php 의 소스코드를 살펴볼 차례입니다.

```
<?
// Send HTTP Header
header('Content-Type:application/javascript');

// Connect to DB
$dbi = mysql_connect('localhost', '****', '****');
mysql_select_db('****', $dbi);
mysql_query('SET NAMES utf8', $dbi);

// DB Search
$sql = 'SELECT name FROM member WHERE ';
$sql .= 'name LIKE "%' . mysql_real_escape_string($_GET['s']) . '%"';
$res = mysql_query($sql, $dbi);
$dat = array();

while($v = mysql_fetch_assoc($res))
{
    $dat[] = $v['name'];
}
```

```
// Output
echo $_GET['callback'] . '(' . json_encode($dat) . ')';
?>
```

이 소스코드는 회원명부를 담고 있는 `member` 테이블에서, 이름에 해당하는 `name` 필드에 검색어가 포함된 레코드를 LIKE 문을 사용해 검색합니다. 검색어는 GET 파라메터 s 를 통해 입력받습니다. 검색한 결과는 PHP 변수인 $dat 에 배열로 담아두게 됩니다.
주목할 곳은 바로 검색결과를 출력하는 부분에 있습니다.
일반적으로 Ajax 를 사용할 때는 검색결과를 JSON 형식으로 출력하고 마치는 것이 보통입니다. 그러나 JSONP 방식을 사용한 위 예제코드는 조금 달라보입니다. 바로 GET 파라메터 callback 을 통해 전달받은 이름의 함수를 호출하는 자바스크립트 코드를 출력하고 있는 것이죠. 그리고 그 함수의 인자로, 회원명부에서 검색한 결과를 담은 배열이 들어갑니다.

예제코드 실행시의 결과

이제 앞서 살펴본 두 소스코드를 연결해서 생각해보면 JSONP 의 전말이 드러납니다. 일단 검색화면 웹페이지에서 "john"을 입력하고 버튼을 클릭하면, 아래와 같은 <script> 요소가 생성되었을 것입니다.

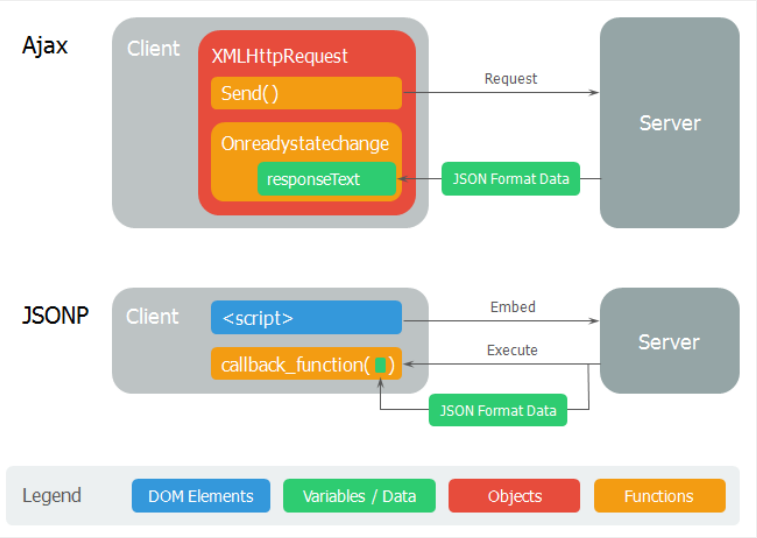
<script src="http://b.epiloum.net/find.php?callback=loadList&s=john"></script>

이렇게 호출된 find.php 파일은 아래와 같이, callback 파라메터로 전달받은 이름인 loadList() 함수를 실행하는 스크립트를 출력합니다. 함수의 인자로 "john"으로 검색한 결과가 배열이 들어오는 것은 물론입니다.
loadList(["John Miller","John C. Potter"])
이렇게 검색화면 웹페이지에서 정의한 loadList() 함수가 실행되면, 처음 우리가 살펴보았던대로 <ul id="list"> 요소 아래에 John 이라는 단어로 검색한 결과가 목록으로 보이게 됩니다.

JSONP 의 기본원리와 한계

위 예제를 토대로 우리는 이제 JSONP 가 작동하는 원리를 아래와 같이 정리할 수 있습니다.

1. 데이터를 요청할 페이지에, 데이터를 받아 처리할 콜백 함수를 먼저 준비해놓습니다. 그 후에 <script> 요소를 생성하여, 데이터 요청을 합니다.
2. 데이터 요청을 받은 페이지에서는 콜백 함수를 실행하는 스크립트를 출력합니다. 이 때 callback 함수의 인자에는 요청받은 데이터가 들어가게 됩니다.



Ajax 와 JSONP 비교 개념도

우리가 잘 알고 있는 Ajax 와 비교해보면 흐름 자체가 크게 다르지 않습니다. 단지 콜백함수를 실행하는 타이밍이 달라졌을 뿐입니다. Ajax 는 데이터 만을 일단 responseText 속성으로 가져온 후, XMLHttpRequest 객체에서 onreadystatechange 속성에 담긴 콜백함수를 실행합니다. 반면에 JSONP 는 서버측에서 이미 콜백함수를 실행하는 코드를 반환하는 점이 다를 뿐입니다.
이처럼 **Ajax 와 동일한 효과를 가지면서도 동일출처원칙을 회피할 수 있는 것**, 이것이 JSONP 가 가진 효용성이라고 할 수 있습니다.
한편 Ajax 와 비교하여 JSONP 가 가지는 한계도 나타나는데, 바로 **GET Method 만을 사용할 수 있다는 점**입니다. 이것은 JSONP 가 <script> 요소를 사용하기 때문에 가지는 숙명적인 한계라고 할 수 있습니다.

jQuery 에서 JSONP 사용하기

JSONP 의 원리를 설명하기 위하여 Live Javascript 로 예제코드를 작성하였지만, 실무에서는 대부분 jQuery 를 이용하게 될 것입니다. 위 예제파일의 find() 함수를 jQuery 로 변경하면 아래와 같습니다.

```
function find()
{
    var s = encodeURIComponent(document.getElementById('str').value);

    $.ajax({
        dataType: "jsonp",
        url: "http://b.epiloum.net/find.php",
        type: "GET",
        data: {'s':s},
        success: function(data){
```

```
            loadList(data);
        }
    });

    return false;
}
```

jQuery 에서는 위처럼 $.ajax() 스펙을 이용하여 평소에 Ajax 를 구현하던 것과 동일한 모양으로 JSONP 를 구현할 수 있습니다. Ajax 와 다른 점은 단지 dataType 값을 "jsonp"로 놓았다는 것 뿐입니다.

다만 이렇게 **JSONP 를 구현할 경우, type 값은 무조건 GET 으로 고정**해야 합니다. POST 등으로 설정할 경우, data 값에서 설정한 파라메터들이 실제로 전달되지 않아 정상작동하지 않는 경우가 있습니다. 아울러 JSONP 를 사용할 경우에는 통신실패시 처리함수인 error 값은 사용할 수 없음을 유념해야 할 것입니다.

JSON (JS Object Notation)

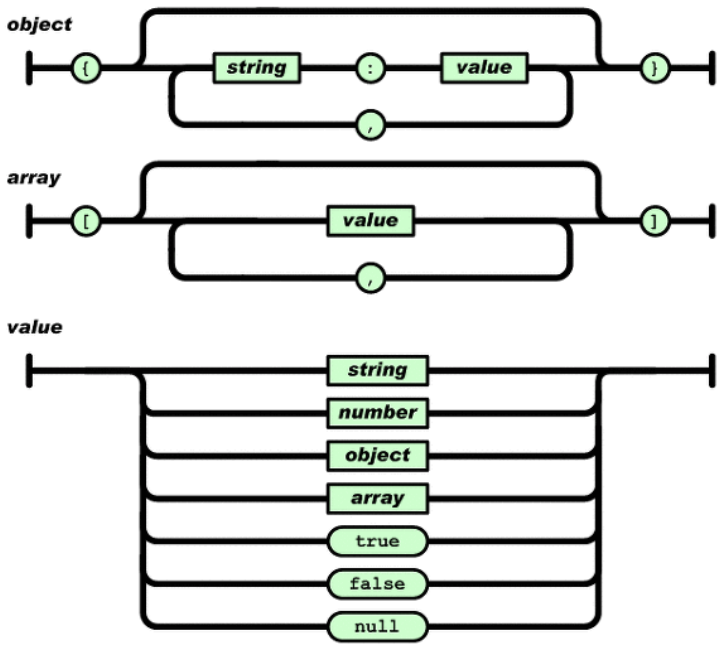
1. JSON(JavaScript Object Notation)

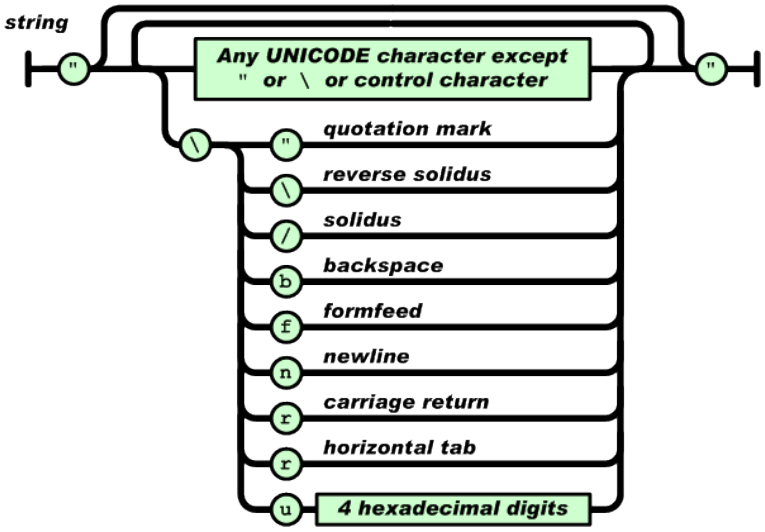
- 경량(Lightweight)의 data 교환 형식임
- 사람이 읽고 쓰기에 용이하며, 기계가 분석하고 생성함에도 용이함

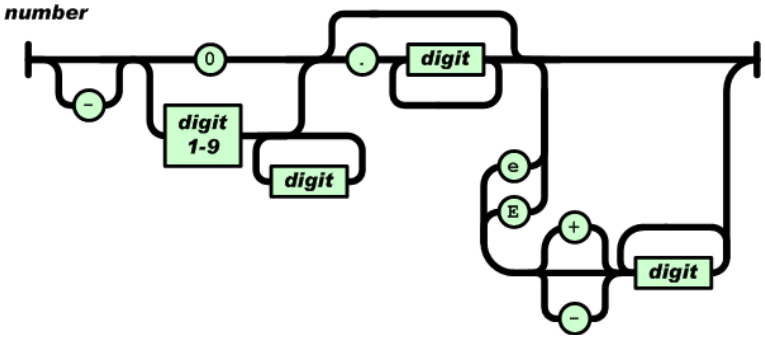
2. JSON 의 구조

- 이름/값의 쌍으로 된 묶음으로 object, record, struct, dictionary, hash table, keyed list, associative array 로 실현됨
- 정렬된 값들의 리스트로 대부분의 언어에서 array, vector, list, sequence 로 실현됨

3. JSON 의 형식







4. JSON 의 이점

- JSON 객체에는 자료형이 부여되므로 별도의 타입 시스템을 만들 필요가 없음
- 데이타를 파싱할 필요가 없음. name/value 의 간단한 구조의 형태로 표현된 데이타를 responseText 로 받아 별도의 파서없이 사용하면 됨
- JSON 은 매우 유연하기도 하며 데이타들의 이름을 교체해서 사용할 수도 있음
- 텍스트 형태이기 때문에 매우 빨리 처리함
- <script> 태그의 src attribute 에 서버의 url 을 명시하기만 하면 어떤 서버에서도 데이타를 가져올 수 있음

*** JSON (JavaScript Object Notation)**

- 데이터를 저장하고 표현할 목적으로 사용되는 것
- 언어가 달라도 모두 사용 가능
- XML 은 정보를 저장하기 위한 불필요 공간이 필요한데 반해서 **JSON 은 최소한의 정보를 저장한다.**

= <name> 황의진 </name>

<name> 황의진

= "name" : "황의진"

키 값

- 배열로 사용가능하고, 객체로도 사용가능하다.
- **ECMAScript 262 의 문법을 따른다.**
- JavaScript 에서 사용할때에는 eval() 함수를 가지고 객체로 만들어서 사용.
  eval(" ' "+responseText|" ' ")
- JSON 문법 (단순 객체로 저장할 때)
  { 키 : 값, 키 : 값 }

  ┌JSON 객체 ↓
var a = eval( );
  └ a.키

var info = eval(" ' {name:황,addr:오류} ' ");
info.name info.addr
황 오류

*** JSON 배열**

[ ] - 값의 배열

{ 키: [값, ...], ... };

var info = eval(" ' {name:[황의진, 천재] , addr:오류동} ' ");
info.name[0]
황의진
info.name[1]
천재

[ ] - JSON Object 의 배열

[{키, 값}, {키, 값} ...];

var info = eval (" ' [{name:황,addr:오류}, {name:임, addr:안양}] ' ")
info[0].name // 황
info[1].name // 임

JSON - 간단한 예제 (HTML with JavaScript)

books 의 정보는 title, author, isbn 으로 되어 있습니다.
JSON 으로 해당정보를 얻으려면 아래와 같이 사용하면 됩니다.

```
<html>
    <head>
```

```
<title>JSON Test</title>
<script type="text/javascript" language="JavaScript">
function jsonTest()
{
var jsonData = {"books": [
        {"title":"Hyperion", "author":"Dan Simmons", "isbn":"0553283685"},
        {"title":"The Stars My Destination", "author":"Alfred Bester", "isbn":"0679767800"},
        {"title":"Black House", "author": ["Stephen King", "Peter Straub"],"isbn":"0345441036"},
        {"title":"The Golden Compass", "author":"Philip Pullman", "isbn":"0679879242"},
]};

alert(jsonData.books[1].author); // Alfred Bester
alert(jsonData.books[3].isbn); // 0679879242
alert(jsonData.books[2].isbn); // 0345441036
alert(jsonData.books[0].author); // Dan Simmons
alert(jsonData.books[2].title); // Black House
alert(jsonData.books[2].author[1]); // Peter Straub
}
</script>
</head>

<body>
        <input type="button" value="Test" name="Test" onClick="jsonTest();" />
</body>
</html>
```

JSON JSP 와 Ajax 예제

Example 1 - Server side JSP encoding

service.jsp :

```
<%@page contentType="text/html; charset=UTF-8"%>
<%@page import="org.json.simple.JSONObject"%>
<%
  JSONObject obj=new JSONObject();
  obj.put("name","foo");
  obj.put("num",new Integer(100));
  obj.put("balance",new Double(1000.21));
```

```
    obj.put("is_vip",new Boolean(true));
    obj.put("nickname",null);
    out.print(obj);
    out.flush();
  %>
```

Please note that you need to place json_simple-1.1.jar in WEB-INF/lib before running the JSP. Then the client side will get the resulting JSON text.

Example 2 - Client side XMLHttpRequest

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>

<script type="text/javascript">
function createXMLHttpRequest(){
  // See http://en.wikipedia.org/wiki/XMLHttpRequest
  // Provide the XMLHttpRequest class for IE 5.x-6.x:
  if( typeof XMLHttpRequest == "undefined" ) XMLHttpRequest = function() {
    try { return new ActiveXObject("Msxml2.XMLHTTP.6.0") } catch(e) {}
    try { return new ActiveXObject("Msxml2.XMLHTTP.3.0") } catch(e) {}
    try { return new ActiveXObject("Msxml2.XMLHTTP") } catch(e) {}
    try { return new ActiveXObject("Microsoft.XMLHTTP") } catch(e) {}
    throw new Error( "This browser does not support XMLHttpRequest." )
  };
  return new XMLHttpRequest();
}

var AJAX = createXMLHttpRequest();

function handler() {
  if(AJAX.readyState == 4 && AJAX.status == 200) {
      var json = eval('(' + AJAX.responseText +')');
      alert('Success. Result: name => ' + json.name + ',' + 'balance => ' + json.balance);
  }else if (AJAX.readyState == 4 && AJAX.status != 200) {
    alert('Something went wrong...');
  }
```

```
}

function show(){
   AJAX.onreadystatechange = handler;
   AJAX.open("GET", "service.jsp");
   AJAX.send("");
};
</script>

<body>
   <a href="#" onclick="javascript:show();"> Click here to get JSON data from the server side</a>
</html>
```

Please place client.html and service.jsp (see Example 1) in the same directory and then open client.html in IE or Firefox, click the link and you'll get result.

JSON(JS Object Notation) 문자열을 객체로 전환

JSON(Javascript Object Notation)은 경량 데이터 교환 형식이다.
사람이 읽기 쉽고, 쓰기 쉽고, 또 기계에게도 해석과 생성이 용이한 형식이며, JavaScript(ECMAScript)에 근거한 부분집합이다.
즉, 자바스크립트로 객체를 기술하는 방법이다.

예들 들어, 아래와 같이 씁니다.

```
var oj = {
      "프로퍼티 이름" : "값",
      "메소드 이름" : function() {alert('This is method')}
}
```

이것만으로 오브젝트 oj 를 만드는 것이 가능하여, **oj.프로퍼티이름** 으로 값을 얻어 낼 수 있어, **oj.메소드이름()** 으로 "This is method"라는 대화창을 표시합니다.
또한 Javascript 로는 배열도 다음과 같은 꼴로 다룰 수 있는 경우가 자주 있습니다.

```
var ary = ["값 1", "값 2", "값 3"]
```

데이터 교환용 형식으로 고려해보면, 이것은 XML 과 마찬가지 또는 그 이상으로 가벼우며 간단한 형식이라는 것을 알 수 있습니다.

특히 웹 상에 보급하고 있는 언어는 C 언어계가 많아, C, C++, C#, 자바, JavaScript, Perl, Python 등 많은 언어간의 연동을 꾀하기 쉬운 형식이라고 할 수 있겠습니다.
그래서 XML 을 대신 할 데이터 교환용 형식으로 Ajax 에서는 JSON 이 보급되고 있습니다.
키가 없는 배열에 대해서는 []를 사용한다.
KOR 은 obj[0]으로 접근할 수 있고 CHN 은 obj[2]로 접근할 수 있다.

```
var obj = ["KOR","USA","CHN"];
```

키가 있는 배열에 대해선 {키:값}을 사용한다.

```
var man = {name:"홍길동",height:"170"};
var result = {
	mans : [
		{name:"홍길동",height:"170"},
		{name:"홍길순",height:"165"}
	]
}
```

[접근법]

```
result.mans[0].name; //홍길동
result.mans[1].name; //홍길순
```

XML 이 커질수록 XML DOM 을 이용하여 객체로 표현하는 것은 매우 수고스러운 작업이 될 것이다.
JSON(제이슨)은 이러한 단점을 보완할 수 있는 표기법으로 XML 대신 JSON 으로 표기된 문자열은 자바스크립트에서 쉽게 객체로 복구하여 사용할 수 있다.

```
var json_str = "{name:'홍길동',height:'170'}";
var man = eval("(" + json_str + ")");
alert(man.name);
```

따라서 서버의 응답이 JSON 문자열이라면 XML DOM 을 만들 필요도 없다.
아주 간단히 결과를 객체화하여 사용할 수 있다.

JSONArray 를 Servlet 에서 받는 방법 (한글깨짐 문제 해결방법)

다른 곳에서 JSONArray 로 데이터를 전송하고, Servlet 에서 이를 받아서 parsing 을 하고 수신여부를 json 으로 전달해야 하는 경우가 발생했다면.

[송신측 코드]

```
public static void main(String[] args) throws UnsupportedEncodingException
{
	HttpURLConnection conn = null;
	URL url = null;
	PrintWriter postReq = null;
	BufferedReader postRes = null;
	String requestJson = null;
	String resultJson = null;
	StringBuilder resultBuffer = new StringBuilder();
	int jsonRtMsg = 1;
	ArrayList<Map> list = new ArrayList<Map>();
	Map<String, String> params = new HashMap<String, String>();
	params = new HashMap<String, String>();
	params.put("id", "20000134");
	params.put("sub_id", "1");
	params.put("path", "/target/대한민국 2.mp4");
	params.put("target", null);
	list.add(params);
	params = new HashMap<String, String>();
	params.put("id", "20000135");
	params.put("sub_id", "1");
	params.put("path", "/target/대한민국.mp4");
	params.put("target", null);
	list.add(params);
	requestJson = JSONArray.fromObject(list).toString();
	System.out.println("[ 요청 JSONArray : " + requestJson + " ]");
	try
	{
		url = new URL("http://abc.com");
		conn = (HttpURLConnection) url.openConnection();
		// 헤더 설정
		conn.setDoOutput(true);
		conn.setDoInput(true);
		conn.setUseCaches(false);
		conn.setRequestMethod("POST");
		conn.setRequestProperty("Content-Type", "application/json;");
		conn.setRequestProperty("Content-Length", Integer.toString(requestJson.length()));
```

```
            conn.setRequestProperty("Accept", "application/json");
            // JSONArray 전송
            postReq  = new PrintWriter(new OutputStreamWriter(conn.getOutputStream(), "UTF-8"));
            postReq.write(requestJson);
            postReq.flush();
            // JSONObject 수신
            postRes = new BufferedReader(new InputStreamReader(conn.getInputStream()));
            while ((resultJson = postRes.readLine()) != null)
            {
                  resultBuffer.append(resultJson);
            }
            conn.disconnect();
            JSONObject json = JSONObject.fromObject(resultBuffer.toString());
            jsonRtMsg = Integer.parseInt(json.getString("resultCode"));
            System.out.println("요청 결과 [" + json.toString() + "]");
      }
      catch (Exception e) {
            System.out.println("[ 요청 실패] ");
      }
}
```

먼저 request 객체로부터 전달된 데이터를 얻어서 JSONArray 로 변환한다.

[수신측 코드]

```
public void requestOrder(HttpServletRequest req, HttpServletResponse
res) throwsUnsupportedEncodingException
{
      ArrayList<Map> list = new ArrayList<Map>();
      String str;
      StringBuilder sb = new StringBuilder();
      try
      {
            BufferedReader br
            = new BufferedReader(newInputStreamReader(req.getInputStream(), "UTF-8"));
            while ((str = br.readLine()) != null) { sb.append(str); }
            arrayObj = JSONArray.fromObject(sb.toString());
      }
```

```
        catch (IOException e)
        {
            e.printStackTrace();
        }

        String json_message = null;
        for (int i = 0; i < arrayObj.size(); i++)
        {
            JSONObject jsonObject = arrayObj.getJSONObject(i);
            id = jsonObject.getString("id");
            sub_id = jsonObject.getString("sub_id");
            path = new String(jsonObject.getString("path").getBytes("8859_1"), "UTF-8");
            target = jsonObject.getString("target");
            jsonObject.put("path", path_prefix + path);
            jsonObject.put("seller_url", return_url);
        }
        try
        {
            res.setCharacterEncoding("utf-8");
            if (message == null)
                res.getWriter().write(makeResultMessageForJSON(0, message));
            else
                res.getWriter().write(makeResultMessageForJSON(1000, message));
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
}
```

한글이 들어 있으면 위의 예처럼 아래 코드를 사용하면 올바르게 받을 수 있다.

```
new String(jsonObject.getString("path").getBytes("8859_1"), "UTF-8");
```

```
jsonObject.put("path", path_prefix + path);
```

이 코드는 path 라는 변수에 path_prefix + path 값을 할당한다.
기존 변수에 새로운 값을 넣거나 새로운 변수에 값을 넣을 때 사용하면 된다.

마지막으로 제일 중요한 것은 송신측에서 값을 null 로 설정해서 보내면 수신측에서는 null 이 아니라 "null" 이라는 String 값으로 받는다.

jQuery ajax 가이드

개요

jQuery 는 브라우저 호환성이 있는 다양한 기능을 제공하는 자바스크립트 라이브러리이다. jQuery 에서 제공하는 오픈 라이브러리들을 통해 java script 로 ajax, event, 다양한 ui 기능 등을 구현할 수 있으며 여기에서는 jQuery 의 기본적인 몇가지 기능(ajax, callback 함수, post 호출 등)에 대하여 살펴본다.
자세한 내용은 jQuery 싸이트를 살펴보도록 한다.
jQuery ajax 의 다양한 기능들 중 기본 Ajax 기능과 응용을 통한 콤보박스, Select 박스의 간단한 화면처리에 대하여 설명한다.

- jQuery ajax 기본기능

- jQuery.ajax()
- jQuery.get()
- jQuery.post()

- jQuery ajax 응용

- Select box
- Tabs

설정

jQuery 를 이용하기 위해서는 jQuery javascript 를 추가해주어야 한다. 추가하는 방법은 jquery url 을 직접 명시하는 경우, 프로젝트에 jquery java script 를 직접 추가하여 참조하는 경우가 있다.

- jQuery url 을 직접 명시하는 경우

```
<script src="//code.jquery.com/jquery-1.11.0.min.js"></script>
```

- jQuery script 를 직접 추가하여 참조하는 경우

```
<script type="text/javascript" src="jQuery 파일 경로"></script>
```

jquery-버전.js 를 다운받아 프로젝트 하위경로에 추가한 후, 저장한 경로를 적어준다.

jQuery AJAX 의 기본 기능

jQuery.ajax()

jQuery Ajax 기능을 위해서는 기본적으로 **jQuery.ajax(url[,settings])** 함수를 이용한다.

ajax함수 안의 settings

jQuery ajax 함수 안에는 다음과 같은 설정들이 가능하다.

설정	설명	default	type
url	request를 전달할 url명	N/A	url string
data	request에 담아 전달할 data명과 data값	N/A	String/Plain Object/Array
contentType	server로 데이터를 전달할 때 contentType	'application/x-www-form-urlencoded; charset=UTF-8'	contentType String
dataType	서버로부터 전달받을 데이터 타입	xml, json, script, or html	xml/html/script/json/jsonp/multiple, space-separated values
statusCode	HTTP 상태코드에 따라 분기처리되는 함수	N/A	상태코드로 분리되는 함수
beforeSend	request가 서버로 전달되기 전에 호출되는 콜백함수	N/A	Function(jqXHR jqXHR, PlainObject settings)
error	요청을 실패할 경우 호출되는 함수	N/A	Function(jqXHR jqXHR, String textStatus, String errorThrown)
success	요청에 성공할 경우 호출되는 함수	N/A	Function(PlainObject data, String textStatus, jqXHR jqXHR)
crossDomain	crossDomain request(jsonP와 같은)를 강제할 때 설정 (cross-domain request 설정 필요)	same-domain request에서 false, cross-domain request에서는 true	Boolean

예제 1

하나의 파라미터를 ajax request 로 전달하는 예제는 다음과 같다.

example01.do 로 호출을 하며 sampleInput 이란 데이터명으로 "sampleData" String 을 전달한다. 요청이 성공할 경우 success 의 함수를 호출하며 실패시 error 함수를 호출하는 자바스크립트 코드이다.

```
$.ajax({
   url : "<c:url value='/example01.do'/>",
   data : {
      sampleInput : "sampleData"
```

```
  },
  success : function(data, textStatus, jqXHR) {
    //Sucess 시, 처리
  },
  error : function(jqXHR, textStatus, errorThrown){
    //Error 시, 처리
}
});
```

ajax함수의 callback함수

ajax 의 콜백함수이다. jQuery 1.5 부터 jQuery 의 모든 Ajax 함수는 XMLHttpRequest 객체의 상위 집합을 리턴받을 수 있게 되었다. 이 객체를 jQuery 에서는 jqXHR 이라 부르며, jqXHR 의 함수로 콜백함수를 정의할 수 있다.

아래 콜백함수와 위의 settings 에서 정의된 error, success 콜백함수와 다른 점은 다음과 같다.

- 사용자 정의에 의해 순차적으로 실행된다.
- ajax 에서 request 를 리턴받아 호출할 수 있다.

함수명	설명
jqXHR.done(function(data, textStatus, jqXHR) {});	성공시 호출되는 콜백함수
jqXHR.fail(function(jqXHR, textStatus, errorThrown) {});	실패시 호출되는 콜백함수
jqXHR.always(function(data|jqXHR, textStatus, jqXHR'|errorThrown) { });	항상 호출되는 콜백함수

예제 2

여러 개의 데이터를 전달하며 호출 후 콜백함수로 서버에서 값을 받는 예제이다.

example02.do 을 호출하며 name, location 을 요청데이터로 전달한다. 성공시에 done 콜백함수를 호출한다.

```
$.ajax({
  url : "<c:url value='/example02.do'/>",
  data : {
    name : "gil-dong",
    location : "seoul"
  },
})
  .done(function( data ) {
    if ( console && console.log ) {
```

```
        console.log( "Sample of data:", data.slice( 0, 100 ) );
    }
});
```

예제 3

example03.do 를 호출하며 성공시 done 콜백함수를, 실패시 fail 콜백함수가 호출된다.
성공, 실패여부에 상관없이 always 콜백함수는 항상 호출된다. done, fail, always 콜백함수는 ajax 함수를 통해 리턴되는 request 로 호출 가능하다.

```
var jqxhr = $.ajax( "<c:url value='/example03.do'/>", )
  .done(function() {
    alert( "success" );
  })
  .fail(function() {
    alert( "error" );
  })
  .always(function() {
    alert( "complete" );
  });

jqxhr.always(function() {
  alert( "second complete" );
});
```

jQuery.get()

jQuery 1.5 부터 sucess 콜백함수는 jqXHR(XMLHttpRequest 의 상위집합 객체)를 받을 수 있게 되었다. 그러나 JSONP 와 같은 cross-domain request 의 GET 요청 시에는 jqXHR 을 사용하여도 XHR 인자는 success 함수 안에서 undefined 로 인식된다.
jQuery.get()은 ajax 를 GET 요청하는 함수이며 jqXHR 을 반환받는다. 따라서 $.ajax()와 동일하게 done, fail, always 콜백함수를 쓸 수 있다.

get 함수는 ajax 함수로 나타내면 다음과 같다.

```
  $.ajax({
    url: url,
    data: data,
    success: success,
    dataType: dataType
  });
```

get함수 설정

설정	설명	default	type
url	request를 전달할 url명	N/A	url String
data	request에 담아 전달할 data명과 data값	N/A	String/Plain Object
dataType	서버로부터 전달받을 데이터 타입	xml, json, script, or html	String
success	요청에 성공할 경우 호출되는 함수	N/A	Function(PlainObject data, String textStatus, jqXHR jqXHR)

예제 1

url 만 호출하고 결과값은 무시하는 경우

```
$.get( "example.do" );
```

예제 2

url 로 데이터만 보내고 결과는 무시하는 경우

```
$.get( "example.do", { name: "gil-dong", location: "seoul" } );
```

예제 3

url 를 호출하고 결과값을 Alert 창으로 띄우는 경우

```
$.get( "test.php", function( data ) {
  alert( "Data Loaded: " + data );
});
```

예제 4

url 를 호출하고 결과값을 Alert 창으로 띄우는 경우

```
$.get( "example.do", { name: "gil-dong", location: "seoul" } )
  .done(function( data ) {
    alert( "Data Loaded: " + data );
});
```

jQuery.getJSON()

ajax 호출을 HTTP GET 메서드로 JSON 문자열로 인코딩한 데이터를 요청한다.

$.ajax()메서드로 표현하면 다음과 같다.

```
$.ajax({
  url: url,
```

```
    data: data,
    success: success,
    dataType: 'json'
});
```

getJSON함수 설정

설정	설명	default	type
url	request를 전달할 url명	N/A	url String
data	request에 담아 전달할 data명과 data값	N/A	String/Plain Object
dataType	서버로부터 전달받을 데이터 타입	xml, json, script, or html	String

jQuery.post()

jQuery.post()은 ajax 를 POST 요청하는 함수이며 jqXHR 을 반환받는다. 따라서 ajax(), get()와 동일하게 done, fail, always 콜백함수를 쓸 수 있다.
jQuery.post 함수 설정은 get 함수와 동일하다.(jQuery.post(url [, data] [, success] [, dataType]))

jQuery.post 함수를 ajax 함수로 쓰면 다음과 같다.

```
$.ajax({
  type: "POST",
  url: url,
  data: data,
  success: success,
  dataType: dataType
});
```

예제 1

url 만 호출하고 결과값은 무시하는 경우

```
$.post( "example.do" );
```

예제 2

url 로 데이터만 보내고 결과는 무시하는 경우

```
$.post( "example.do", { name: "gil-dong", location: "seoul" } );
```

예제 3

url 를 호출하고 결과값을 console log 를 남기는 경우

```
$.post( "example.do", function( data ) {
```

```
    console.log( data.name );
    console.log( data.location );
});
```

예제 4

url 로 데이터를 호출하고 결과값을 Alert 창으로 띄우는 경우

```
$.post( "example.do", { name: "gil-dong", location: "seoul" } )
    .done(function( data ) {
        alert( "Data Loaded: " + data );
});
```

jQuery Ajax 응용

jQuery 의 ajax 추가 UI 기능(자동완성기능, 판넬 탭 등)을 사용하기 위해서는 jQuery UI 를 설정해주어야 한다. 여기에서는 jQuery UI 와 ajax 를 이용한 자동완성기능(autocomplete), 판넬 탭(tabs)에 대하여 가이드한다.

jQuery UI 설정

jQuery UI(version 1.11.0)을 추가하기 위해서는 위에서 설정했던 기본 jQuery script 와 jQuery ui 스크립트를 다음과 같이 jsp 에 추가해준다.

```
...
<link rel="stylesheet"
        href="http://code.jquery.com/ui/1.11.0/themes/smoothness/jquery-ui.css" />
<script src="//code.jquery.com/jquery-1.11.0.min.js"></script>
<script src="http://code.jquery.com/ui/1.11.0/jquery-ui.js"></script>
...
```

jQuery UI script 를 직접 추가하여 참조하는 경우는 jQuery-ui.js 와 jQuery-ui.css 를 다운받아 프로젝트 하위 경로에 추가한 후, 저장한 경로를 지정해준다.

Auto complete

jQuery 에서는 input 창에서 예상되는 텍스트값을 보여주는 자동완성기능을 쉽게 구현할 수 있도록 autoComplete()을 제공하고 있다.

autoComplete의 설정

구분	설정	설명	Type
Options	source	하단에 뜨는 자동완성리스트(필수값)	Array, String, function
Options	minLength	자동완성이 동작하는 최소 문자열 수	Integer
Options	disabled	disable 여부	Boolean
Events	change(event, ui)	값 변경시 발생하는 이벤트 함수	autocompletechange
Events	focus(event, ui)	값이 포커스될 때 발생하는 이벤트 함수	autocompletefocus
Events	select(event, ui)	값이 선택될 때 발생하는 이벤트 함수	autocompleteselect

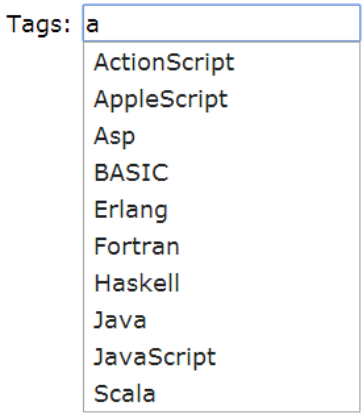
자세한 내용은 jQuery 싸이트의 autocomplete api 을 참고한다.

autoComplete 기본 예제

기본 autoComplete 기능 구현은 다음과 같다.

```
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI Autocomplete - Default functionality</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.11.0/themes/smoothness/jquery-ui.css">
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.11.0/jquery-ui.js"></script>
  <link rel="stylesheet" href="/resources/demos/style.css">
  <script>
  $(function() {
    var availableTags = [ "ActionScript", "AppleScript", "Asp", "BASIC",
                          "C", "C++", "Clojure", "COBOL", "ColdFusion", "Erlang",
                          "Fortran", "Groovy", "Haskell", "Java", "JavaScript", "Lisp",
                          "Perl", "PHP", "Python", "Ruby", "Scala", "Scheme" ];
    $( "#tags" ).autocomplete({
      source: availableTags
    });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label for="tags">Tags: </label>
  <input id="tags">
</div>
</body>
</html>
```

위와 같이 했을 때 결과는 다음과 같다.



minLength 는 default 값이 1 이기 때문에 input 에 1 개이상의 문자를 입력했을 때 source 의 String 배열들이 자동문자리스트로 뜨게 된다.

autoComplete와 ajax를 이용한 응용예제

ajax 를 통해 리스트를 받아와 autoComplete 의 source 로 뿌려주는 예제에 대해 살펴보자.

ajax 를 통해 source 를 가져오기 위해서는 서버호출 결과값이 2 가지 중 하나의 타입이어야 한다.

- String array 타입
- Object(id, label,value 값을 갖는) array 타입

1. String array로 가져오는 경우

example.do 라는 url 로 ajax 호출을 통해 source 를 가져오고 선택 시 값이 alert 되도록 하는 예제이다.

<javascript>

```
...
<script type="text/javaScript">
$(function() {
    $('#autoValue').autocomplete(
        {
            source : function(request, response) {
                $.ajax({
                    url : "<c:url value='/example.do'/>",
                    data : { input : request.term },
                  success : function(data) {
                        response( data.locations );
                    }
                });
            },
              minLength : 1,
```

```
            select : function(event, ui) {
               alert( ui.item ? "Selected : " + ui.item.label
                        : "Nothing select, input was " + this.value);
            }
      });
});
</script>
//... 생략
<input type="text" id="autoValue" />
//... 생략
```

data 로 전송하는 request.term 은 input 값으로 사용자가 입력한 값이다. 즉, 사용자가 "je"를 입력하면 input = je 로 값이 넘어간다.
이 때 Controller 에서는 reqeust.getParameter("input") 또는 @RequestParam("input") String input 으로 값을 꺼낼 수 있다.
다음은 MappingJacksonJsonView 의 빈을 jsonView 로 등록했을 때 Controller 에서 data 를 꺼내고 결과값을 client 로 넘겨주는 예제이다.(Ajax 통신 시 java 코드는 <Ajax support java code 위키>를 참고한다.)

```
 @RequestMapping(value="/autoList.do")
public String autoList(HttpServletRequest request, ModelMap model) {

            String input = request.getParameter("input");
            List<String> resultList = new ArrayList<String>();
           //...생략...
            //서비스클래스를 통해 결과값을 resultList 에 담음
            model.addAttribute("locations", resultList );

            return "jsonView";
}
```

이 때, 호출되는 Query 문의 예이다. (mybatis 예제)

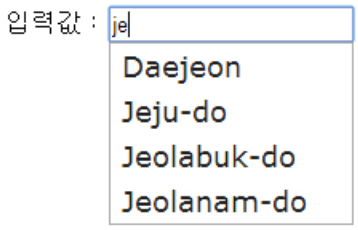
```
<select id="selectLocationList" parameterType="string" resultType="string">
SELECT
            LOCATION_NM
FROM LOCATION
            WHERE upper(LOCATION_NM) LIKE '%' || upper(#{input}) || '%'
```

```
</select>
```

만약 결과값이 다음과 같다면

```
{"locations":["Daejeon","Jeju-do","Jeolabuk-do","Jeolanam-do"]}
```

ajax 의 성공시 콜백함수인 success 에서는 data.locations 로 값을 꺼내 autocomplete 의 source 를 설정할 수 있다.
위와 같은 경우 결과 화면은 다음과 같다.



2. Object array로 가져오는 경우

위와 동일하게 Query 문을 통해 입력값에 따라 데이터를 검색하고 Object array 로 서버에서 결과값을 가져오는 경우에 대해 살펴본다.

Controller 에서 List<Object>의 값을 ModelMap 에 "locations"라는 이름으로 클라이언트로 넘겨주고 서버에서 다음과 같이 json data 가 넘어왔을 때

```
{"locations":[{"locationId":"0006","locationNm":"Daejeon","localNb":"042"},{"locationId":"0010","locationNm":"Jeju-do","localNb":"064"},{"locationId":"0011","locationNm":"Jeolabuk-do","localNb":"063"},{"locationId":"0012","locationNm":"Jeolanam-do","localNb":"061"}]}
```

autocomplete 의 source 에 넘어온 값이 나타나도록 하는 Object 는 label, id, value 값을 가질 수 있다. 그렇기 때문에 넘어온 request 의 값을 success 콜백함수에서 source 에 나타나도록 하는 Object 형태로 변환해야 한다.

나머지 jQuery 구현은 동일하다.

```
success : function(data) {
            response($.map(data.locations, function(item) {
                        return{
                                    id: item.locationId,
                                    label: item.locationNm,
                                    //value: item.localNb
```

```
                }));
        }
```

이때, 자동완성 리스트로 나타나는 값은 label 이며, value 를 설정해주었을 때는 자동완성 리스트에서 값 선택 시 input 값에 label 값이 아닌 value 값이 대입된다.

Select box

selectbox(combobox) 제어

jQuery 에서는 별도로 select box ui 함수를 제공하지 않는다.
jQuery 를 통해 selectbox 를 제어하는 방법에 대하여 알아보고 selectbox 에 나타나는 리스트를 ajax 로 구현하는 방법에 대하여 살펴본다.

다루고자 하는 selectbox 가 다음과 같다고 가정하자.

```
<select id="combobox">
    <option value="">===locations===</option>
    <option value="01">Seoul</option>
    <option value="02">Busan</option>
    <option value="03">Jeju-do</option>
    <option value="04">Incheon</option>
</select>
```

selectbox 값 가져오기

selectbox 에서 선택된 value 를 가져오는 방법은 다음과 같다.

```
var selectedVal = $("#combobox option:selected").val();
```

selectbox 내용 가져오기

selectbox 에서 선택된 text(ex:Seoul)를 가져오는 방법은 다음과 같다.

```
var selectedText= $("#combobox option:selected").text();
```

selectbox에서 선택된 Index값 가져오기

selectbox 의 리스트에서 선택된 Index 를 구하는 방법은 다음과 같다.

```
var selectedIndex = $("#combobox option").index($("#combobox option:selected"));
```

selectbox에 리스트 마지막에 추가하기

selectbox 의 리스트에 값을 마지막에 추가하는 방법은 다음과 같다.

```
$("#combobox").append("<option value="05">Daejeon</option>");
```

맨 앞에 추가하는 경우는 .prepend()를 쓴다.

selectbox 값 교체하기

selectbox 의 리스트의 값들을 교체하는 방법은 다음과 같다.

```
$("#combobox").html(
    "<option value=''>===locations===</option>
    <option value='01'>Jeju-do</option>
    <option value='02">Seoul</option>
    <option value='03'>Incheon</option>
    <option value='04'>Daejeon</option>")
```

selectbox 값 교체하기

selectbox 의 리스트의 값들을 교체하는 방법은 다음과 같다.

```
$("#combobox").html(
    "<option value=''>===locations===</option>
    <option value='01'>Jeju-do</option>
    <option value='02">Seoul</option>
    <option value='03'>Incheon</option>
    <option value='04'>Daejeon</option>")
```

selectbox에서 값 삭제하기

selectbox 의 리스트에서 선택된 값을 삭제하는 방법은 다음과 같다.

```
$("#combobox option:selected").remove();
```

selectbox에서 값이 선택될 때 콜백함수

selectbox 에서 값이 선택되었을 때 호출되는 콜백함수는 다음과 같다.

```
$("#combobox).change(function() {
    //기능구현
});
```

selectbox 만들기

위에서 쓴 selectbox 제어기능과 jQuery ajax 함수를 이용하여 다음과 같이 간단히 selectbox 를 만들 수 있다.

<JSP 에서 콤보박스 구현 예제>

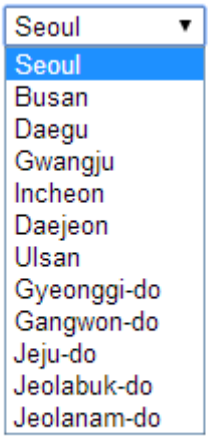
```
<script type="text/javaScript">
		$(function() {
			$.ajax({
				url : "<c:url value='/simpleCombo.do'/>",
				success : function(data) {
					loadCombo($("#combobox"), data.locations);
				$("#combobox").val("");
				}
			});

			$("#combobox").change(function() {
				alert("Selected : " + $("#combobox option:selected").val());
			});
		});

		function loadCombo(target, data) {
			var dataArr = [];
			var inx = 0;
			target.empty();

			$(data).each( function() {
				dataArr[inx++] = "<option value=" + this.locationId + ">" +
this.locationNm + "</option> ";
			});

			target.append(dataArr);
		}
</script>

//... 생략
<select id="combobox">
		<option>===locations===</option>
</select>
```

<서버에서 가져오는 결과값>


```
{"locations":[{"locationId":"0001","locationNm":"Seoul"},{"locationId":"0002","locationNm":"Busan"},{"l
ocationId":"0003","locationNm":"Daegu"},{"locationId":"0004","locationNm":"Gwangju"},{"locationId":
```

"0005","locationNm":"Incheon"},{"locationId":"0006","locationNm":"Daejeon"},{"locationId":"0007","locationNm":"Ulsan"},{"locationId":"0008","locationNm":"Gyeonggi-do"},{"locationId":"0009","locationNm":"Gangwon-do"},{"locationId":"0010","locationNm":"Jeju-do"},{"locationId":"0011","locationNm":"Jeolabuk-do"},{"locationId":"0012","locationNm":"Jeolanam-do"}]}

ajax 함수가 실행되면서 simpleCombo.do 를 통해 data 를 가져오고 json 값이 위와 같을 때 combobox 를 구성하는 함수를 구현하여 combobox 에 나오는 목록을 나타낼 수 있다.
위의 결과는 다음과 같다.



Tabs

Tab 기본 구현하기

jQuery UI 에서 tab 구현은 tabs()를 쓰며 추가 설정은 다음과 같다.

설정	설명	구분
active	활성화될 panel선택	options
event	tab이 활성화되는 이벤트	options
hide	panel이 숨겨질 때 애니메이션	options
show	panel이 나타날 때 애니메이션	options
beforeLoad	Remote tab이 로드되기 전에 실행되기 전에 발생되는 이벤트 함수	events
create		events

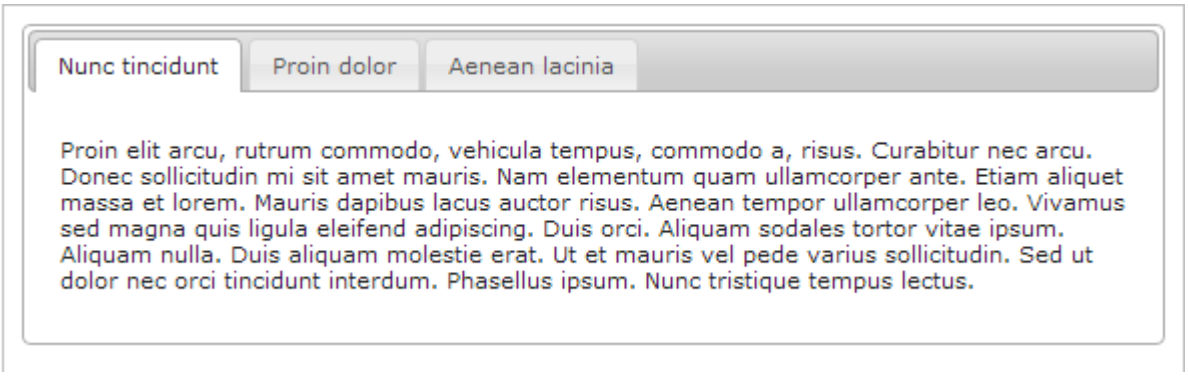
Tab 을 구현하는 경우 기본 예제는 다음과 같다.

```
<!doctype html>
<html lang="en">
```

```
<head>
  <meta charset="utf-8">
  <title>jQuery UI Tabs - Default functionality</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.11.0/themes/smoothness/jquery-ui.css">
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.11.0/jquery-ui.js"></script>
  <link rel="stylesheet" href="/resources/demos/style.css">
  <script>
  $(function() {
    $( "#tabs" ).tabs();
  });
  </script>
</head>
<body>

<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a, risus. Curabitur nec arcu. Donec sollicitudin mi sit amet mauris. Nam elementum quam ullamcorper ante. Etiam aliquet massa et lorem. Mauris dapibus lacus auctor risus. Aenean tempor ullamcorper leo. Vivamus sed magna quis ligula eleifend adipiscing. Duis orci. Aliquam sodales tortor vitae ipsum. Aliquam nulla. Duis aliquam molestie erat. Ut et mauris vel pede varius sollicitudin. Sed ut dolor nec orci tincidunt interdum. Phasellus ipsum. Nunc tristique tempus lectus.</p>
  </div>
  <div id="tabs-2">
    <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus gravida ante, ut pharetra massa metus id nunc. Duis scelerisque molestie turpis. Sed fringilla, massa eget luctus malesuada, metus eros molestie lectus, ut tempus eros massa ut dolor. Aenean aliquet fringilla sem. Suspendisse sed ligula in ligula suscipit aliquam. Praesent in eros vestibulum mi adipiscing adipiscing. Morbi facilisis. Curabitur ornare consequat nunc. Aenean vel metus. Ut posuere viverra nulla. Aliquam erat volutpat. Pellentesque convallis. Maecenas feugiat, tellus pellentesque pretium posuere, felis lorem euismod felis, eu ornare leo nisi vel felis. Mauris consectetur tortor et purus.</p>
  </div>
  <div id="tabs-3">
```

```
    <p>Mauris eleifend est et turpis. Duis id erat. Suspendisse potenti. Aliquam vulputate, pede vel vehicula accumsan, mi neque rutrum erat, eu congue orci lorem eget lorem. Vestibulum non ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Fusce sodales. Quisque eu urna vel enim commodo pellentesque. Praesent eu risus hendrerit ligula tempus pretium. Curabitur lorem enim, pretium nec, feugiat nec, luctus a, lacus.</p>
    <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at, semper at, magna. Nullam ac lacus. Nulla facilisi. Praesent viverra justo vitae neque. Praesent blandit adipiscing velit. Suspendisse potenti. Donec mattis, pede vel pharetra blandit, magna ligula faucibus eros, id euismod lacus dolor eget odio. Nam scelerisque. Donec non libero sed nulla mattis commodo. Ut sagittis. Donec nisi lectus, feugiat porttitor, tempor ac, tempor vitae, pede. Aenean vehicula velit eu tellus interdum rutrum. Maecenas commodo. Pellentesque nec elit. Fusce in lacus. Vivamus a libero vitae lectus hendrerit hendrerit.</p>
  </div>
</div>
</body>
</html>
```

위의 결과는 다음과 같다.



- 참조 : jquery ui site

ajax 로 Tab 구현하기

ajax 로 Tab 을 구현하기 위해서는 각 Tab 에 ajax 호출 url 을 지정해주기만 하면 된다.

```
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.11.0/jquery-ui.js"></script>
<script type="text/javaScript">
$(function() {
    $( "#tabs" ).tabs({
      beforeLoad: function( event, ui ) {
        ui.jqXHR.error(function() {
```

```
            ui.panel.html(
              "Couldn't load this tab. We'll try to fix this as soon as possible. " +
              "If this wouldn't be a demo." );
          });
        }
      });
  });
</script>
</head>
<body>
    <div id="tabs">
        <ul>
            <li><a href="${pageContext.request.contextPath }/simpleCombo.do">Tab 1</a></li>
             <li><a href="${pageContext.request.contextPath }/tabTwoForm.do">Tab 2</a></li>
             <li><a href="${pageContext.request.contextPath }/tabThreeForm.do">Tab 3</a></li>
        </ul>
    </div>
</body>
```

위의 경우 첫번째 탭 지정시 simpleCombo.do 가 호출되어 panel 안에 simpleCombo.do 를 통해 호출된 화면이 보여진다.

자주 쓰이는 ajax 패턴

Ajax 구현할 때는 거의 jQuery 를 활용하고 있습니다. 모바일 웹 구현할 때 네이티브 앱과 같은 UX 를 위해 Ajax 를 많이 사용하게 됩니다.

- **로그인**

로그인은 보통 화면에서 폼 값을 입력한 후 해당 값과 함께 ajax 를 호출합니다.
호출된 액션에서 로그인 관련 인증 및 예외처리 프로세스를 실행 후 인증 상태에 따라 결과 값을 출력합니다.
출력된 값에 따라 로그인 페이지에서 경고를 띄우든 다음 페이지로 진행하든 처리합니다.
ajax 루틴을 돌고 있을 때 사용할 로딩 바를 띄우는 작업도 함께 해주면 좀더 나은 UX 를 구현할 수 있습니다..

login view script

```
<script>
function checkLogin(){

    if( $.trim($("#userId").val()) == '' ){
        alert("아이디를 입력해 주세요.");
        $("#userId").focus();
        return;
    }
    if( $.trim($("#userPw").val()) == '' ){
        alert("비밀번호를 입력해 주세요.");
        $("#userPw").focus();
        return;
    }
    // 로그인 프로세스 호출
    $.ajax({
        type: 'post'
        , async: true
        , url: '/member.do?cmd=login'
        , data: $("#frm").serialize()
        , beforeSend: function() {
            $('#ajax_load_indicator').show().fadeIn('fast');
          }
        , success: function(data) {
            var response = data.trim();
            console.log("success forward : "+response);
```

```
            // 메세지 할당
            switch(response) {
                case "nomatch":
                    msg = "아이디 또는 비밀번호가 일치하지 않습니다."; break;
                case "fail":
                    msg = "로그인에 실패 했습니다."; break;
                default :
                    msg = "존재하지 않는 사용자입니다."; break;
            }
            // 분기 처리
            if(response=="success"){
                window.location.href = "${targetUrl}";
            } else {
                alert(msg);
            }
          }
        , error: function(data, status, err) {
         console.log("error forward : "+data);
            alert('서버와의 통신이 실패했습니다.');
          }
        , complete: function() {
         $('#ajax_load_indicator').fadeOut();
          }
    });
}
</script>
```

login view html

```
<form id="frm" name="frm" method="post" action="" onSubmit="checkLogin();return false;">
    <fieldset>
        <legend>login</legend>
        <div class="login_item mg_top34">
            <label>id</label>
            <input id="userId" name="memberVo.xcWebMbrId" type="text" class="i_login" />
        </div>

        <div class="login_item mg_top10">
            <label>password</label>
            <input id="userPw" name="memberVo.xcPswd" type="password" class="i_login" />
```

```
        </div>

        <div id="ajax_load_indicator" style="display:none">
                        <p style="text-align:center; padding:16px 0 0 0"><img src="/mobile/common/img/ajax-loader-line.gif" /></p>
        </div>

        <p class="keeping mg_left89">
                <input id="keepidpw" class="rd_box22" value="1" type="checkbox" name="idPswdSave" >
            <label for="keepidpw">ID/PW 저장</label>
        </p>

        <p class="keeping mg_left20">
            <input id="keepid" class="rd_box22" value="1" type="checkbox" name="idSave" >
            <label for="keepid">ID 저장</label>
        </p>

        <span class="btn_login">
                <input type="image" src="<%=imageUrl%>/btn/btn_login.jpg" title="로그인" onclick="checkLogin();return false;">
        </span>

                <p class="btn_register"><a href="/member.do?cmd=memberJoin"><img src="<%=imageUrl%>/btn/btn_join.jpg" alt="회원가입" /></a>
                    <a href="/member.do?cmd=goIdPwFind" class="mg_left5"><img src="<%=imageUrl%>/btn/btn_sch.jpg" alt="아이디/비밀번호 찾기" /></a></p>
    </fieldset>
</form>
```

login ajax result

```
<%@ include file="/mobile/common/include/config.jsp" %>
${result}
```

- **내용 추가**

  페이지 하단으로 스크롤 이동했을 때 자동으로 다음 페이지 데이터가 추가되는 UX를 구현할 때 사용하는 기법으로 jQuery 의 append() 함수를 사용할 수 있다.

product list view script

```
<script>
function paging(){
    count++;

    $.ajax({
        type: 'post'
        , async: true
                                                                                , url:
"/display.do?cmd=productListAppend&ordFlag="+'${ordFlag}'+"&categoryCd="+categoryCd+"&it
emCode="+'${itemCode}'+"&count="+count
        , beforeSend: function() {
              $('#ajax_load_indicator').show().fadeIn('fast');
          }
        , success: function(data) {
            var response = data.trim();
            console.log("success forward : "+response);
            // 상품 리스트 할당
            $('#view_list').append(response);
            $('#product_count').html($('#view_list li.thumb').size());
          }
        , error: function(data, status, err) {
            console.log("error forward : "+data);
            alert('서버와의 통신이 실패했습니다.');
          }
        , complete: function() {
            $('#ajax_load_indicator').fadeOut();
          }
    });
}
</script>
```

product list view html

```
<!-- 상품 목록 -->
<div id="view_list" class="product_list" style="display: none;">
<c:forEach var="list" items="${returnMap}">
  <a
href="/display.do?cmd=productView&brandCd=${list.brandCd }&prodCd=${list.prodCd }&ordFlag=
${ordFlag }&styleYY=${list.styleYY }&priceDpYn=${list.priceDpYn }&listPrice=${list.listPrice }&catego
```

```
ryCd=${categoryCd}&itemCode=${itemCode}&rNum=${list.rNum }&count=${count}">
    <ul>
      <li class="thumb"><img src="${list.listImg }" alt="상품이미지" /></li>
                <li class="b_title"><span class="brand">[${list.brandNm }]</span><span
class="title">${list.prodNm }</span></li>
      <li class="price">
      <c:if test="${list.priceDpYn eq 'Y'}">
      ${list.listPrice }원
      </c:if>
      <c:if test="${list.priceDpYn eq 'N'}">
         <span><a href="#"><img src="<%=imageUrl%>/common/ic_login.png" alt="로그인"
/></a></span>
      </c:if>
      </li>
      <li class="btn_go"><img src="<%=imageUrl%>/btn/btn_go.png" alt="" /></li>
    </ul>
  </a>
</c:forEach>
</div>

<div id="ajax_load_indicator" style="display:none">
    <p style="text-align:center; padding:14px 0 14px 0"><img src="/mobile/common/img/ajax-
loader-line.gif" /></p>
</div>

<!-- 더보기 bar-->
<div class="more_bar">
<a href="javascript:paging()">
  <ul class="sec01">
    <li class="btn_arr"><img src="<%=imageUrl%>/common/arr_down.png" alt="" /></li>
    <li class="text_more">15개 더보기</li>
    <li class="text_num"><span id="product_count">${total}</span> / 999</li>
  </ul>
  </a>
  <p><a href="#"><img src="<%=imageUrl%>/btn/btn_top.jpg" alt="맨위로" /></a></p>
</div>
```

append view html

```
<%@ include file="/mobile/common/include/config.jsp"%>
```

```
<!-- 상품 목록 -->

<c:forEach var="list" items="${returnMap}">
                                                                                <a
href="/display.do?cmd=productView&brandCd=${list.brandCd }&prodCd=${list.prodCd }&ordFlag=
${ordFlag }&styleYY=${list.styleYY }&priceDpYn=${list.priceDpYn }&listPrice=${list.listPrice }&catego
ryCd=${categoryCd}&itemCode=${itemCode}&rNum=${list.rNum }&count=${count}">
        <ul>
            <li class="thumb"><img src="${list.listImg }" alt="상품이미지" /></li>
            <li class="b_title"><span class="brand">[HAZZYS]</span><span
                class="title">${list.prodNm }</span></li>
            <li class="price">
            <c:if test="${list.priceDpYn eq 'Y'}">
            ${list.listPrice }원
            </c:if>
            <c:if test="${list.priceDpYn eq 'N'}">
             <span><a  href="#"><img  src="<%=imageUrl%>/common/ic_login.png"  alt="로그인"
/></a></span>
            </c:if>
            </li>
            <li class="btn_go"><img src="<%=imageUrl%>/btn/btn_go.png" alt="" /></li>
        </ul>
    </a>
</c:forEach>
```

ajax 로딩 화면 만들기..

로딩에도 여러가지가 있다.

1. 로딩될 자리에 "로딩중..." 이라는 글자나 뺑글 뺑글 돌아가는 이미지가 바로 뜨는 방법
2. 페이지 젤 하단이나 상단에 "로딩중.. " "저장중.." 이라는 글자가 뜨는 방법
3. 화면 중간에 새창 (레이어)이 떠서 로딩중 글자 뿌려주는 방법 등.

```
<style>
#ajaxBox{
    border:3px solid #000000;
    width:300px;
    height:50px;
    position:absolute;
    background-color:#ffffff;
    text-align:center;
    font-weight:bold;
    padding:20px 0px 0px 0px;
    color:#FF6600;
}
</style>

<div id="ajaxBox"><div id="ajaxBoxMent">잠시만 기다려주세요..</div><img src="/images/ajax-loader.gif"></div>

<script>
function showAjaxBox(ment){

//항상 화면 중앙에 나타나도록...
    var yp=document.body.scrollTop;
    var xp=document.body.scrollLeft;

    var ws=document.body.clientWidth;
    var hs=document.body.clientHeight;

    if(!ment) ment="잠시만 기다려주세요..";

    var ajaxBox=$('ajaxBox');
    $('ajaxBoxMent').innerHTML=ment;
```

```
    ajaxBox.style.top=yp+eval(hs)/2-100;
    ajaxBox.style.left=xp+eval(ws)/2-100;

    Element.show(ajaxBox);
}
</script>
```

사용법은 아래와 같습니다.

```
function test(){

  showAjaxBox();//로딩 박스 보인후에...

    var url="Ajax.php";
    var myAjax=new Ajax.Request(
        url,
        {
            asynchronous:true,
            parameters:pars,
            onComplete:function(res){

                Element.hide('ajaxBox');// 다 처리한후.. 박스를 숨김..

            }
        }
    );
}
```

그리고 페이지 맨 아래쪽에 추가하면, 상자가 처음에는 안보이게 합니다.

```
<script>
Element.hide('ajaxBox');
</script>
```

참고로 **<img src="/images/ajax-loader.gif">** 로딩에 쓸 이미지는 임의대로 정하면 됩니다.

페이지 더보기 클릭시 리스트 불러오기

```
<script>
  var listcount=0;
  function addlist(){
        listcount+=5;
        $.post("/getlist.php",{"count" : listcount},function(data){
              var oldlist =  $("#listbody").html();
              $("#listbody").html(oldlist+data);
        });
  }
</script>

<div id='listbody'>
    <div id='1' >게시글 1</div>
                  .
    <div id='5' >게시글 5</div>                .
</div>
<div onclick='addlist()'> 더보기 </div>
```

처음 리스트가 5 개 있습니다. addlist() 함수를 호출할 때마다 5 씩 플러스 시켜서 getlist.php 에 인자로 넘겨주고 limte 5 씩 증가시켜서 게시물을 쿼리 조회해서 다시 listbody 에 기존 내용 뒤에 붙이는 방법입니다. 줄이는 건 반대로 하면 됩니다.