

[0710]JSON파싱하기 (GSON)

GSON 이란?

🔗 Gson

Gson is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object. Gson can work with arbitrary Java objects including pre-existing objects that you do not have source-code of.

There are a few open-source projects that can convert Java objects to JSON. However, most of them require that you place Java annotations in your classes; something that you can not do if you do not have access to the source-code. Most also do not fully support the use of Java Generics. Gson considers both of these as very important design goals.

JSON을 JAVA객체로 쉽게 바꿔 줄 수 있고, JAVA객체를 JSON 객체로 쉽게 바꿔줄 수 있다.



Gson jar downloads

search.maven.org/artifact/com.google.code.gson/gson/2.8.6/jar

sonatype | Maven Central Repository Search | Quick Stats | Report A Vulnerability | GitHub

com.google.code.gson:gson:2.8.6

le.code.gson:gson: 2.8.6

View on OSS Index

Browse Downloads

Apache Maven
maven.apache.org

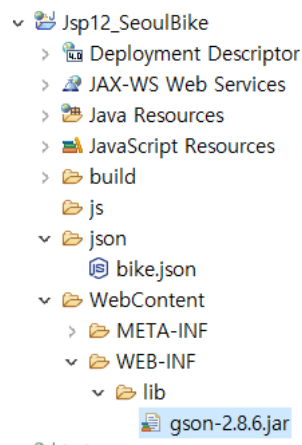
jar
javadoc.jar
pom
sources.jar

<dependency>
<groupId>com.google.code.gson</groupId>
<artifactId>gson</artifactId>
<version>2.8.6</version>
</dependency>

com.google.code.gson:gson
2.8.6

direct.xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www

jar 파일 다운받아서



lib 파일 안에 넣어주기

- bike01.js

```
$(function(){
    getBike();
});

function getBike(){
    $.getJSON("json/bike.json",function(data){
        $.each(data, function(key, val){ /* 반복할수,
            만약 data가 [6,30] 배열형태면 function(index, val) 0 : 6, 1 : 30 이런식으로 리턴
            map { '하나' : '강동원', '둘' : '서강준' } 맵 형태면 하나 : 강동원 , 둘 : 서강준 이런식으로 리턴 */
            if(key == "DESCRIPTION"){
                $("table").attr("border",1);
                $("thead").append(
                    "<tr>"+
                    "<th>"+val.ADDR_GU+"</th>"+ // val = jsonObject
                    "<th>"+val.CONTENT_ID+"</th>"+
                    "<th>"+val.CONTENT_NM+"</th>"+
                    "<th>"+val.NEW_ADDR+"</th>"+
                    "<th>"+val.CRADLE_COUNT+"</th>"+
                    "<th>"+val.LONGITUDE+"</th>"+
                    "<th>"+val.LATITUDE+"</th>"+
                    "</tr>"+
                );
            }
            else {
                var list = val;
            }
        });
    });
}
```

```


for(var i=0; i<list.length; i++){
    var str = list[i];
    $("tbody").append(
        "<tr>" +
        "<td>" + str.addr_gu + "</td>" +
        "<td>" + str.content_id + "</td>" +
        "<td>" + str.content_nm + "</td>" +
        "<td>" + str.new_addr + "</td>" +
        "<td>" + str.cradle_count + "</td>" +
        "<td>" + str.longitude + "</td>" +
        "<td>" + str.latitude +
        "<input type='hidden' name='bike' value='" +
        str.addr_gu + "/" +
        str.content_id + "/" +
        str.content_nm + "/" +
        str.new_addr + "/" +
        str.cradle_count + "/" +
        str.longitude + "/" +
        str.latitude + "' />" +
        "</td>" +
        "</tr>"
    );
}
});
});
}

```

\$.each() 와

.each()는 전혀 다른 함수 이다.

.each(function) : 한 줄씩 진행되는 함수

.each(function)	Returns: jQuery
Description: <i>Iterate over a jQuery object, executing a function for each matched element.</i>	
 .each(function)	version added: 1.0
function Type: Function (Integer index, Element element) A function to execute for each matched element.	
<p>The <code>.each()</code> method is designed to make DOM looping constructs concise and less error-prone. When called it iterates over the DOM elements that are part of the jQuery object. Each time the callback runs, it is passed the current loop iteration, beginning from 0. More importantly, the callback is fired in the context of the current DOM element, so the keyword <code>this</code> refers to the element.</p>	

\$.each() : 반복함수 (배열이나 map 형태일때)

Description: A generic iterator function, which can be used to seamlessly iterate over both objects and arrays. Arrays and array-like objects with a length property (such as a function's arguments object) are iterated by numeric index, from 0 to length-1. Other objects are iterated via their named properties.

jQuery.each(array, callback)

version added: 1.0

array

Type: [ArrayLikeObject](#)

The array or array-like object to iterate over.

callback

Type: [Function](#)([Integer](#) indexInArray, [Object](#) value)

The function that will be executed on every value.

jQuery.each(object, callback)

version added: 1.0

object

Type: [Object](#)

The object to iterate over.

callback

Type: [Function](#)([String](#) propertyName, [Object](#) valueOfProperty)

The function that will be executed on every value.

The `$.each()` function is not the same as `$(selector).each()`, which is used to iterate, exclusively, over a jQuery object. The `$.each()` function can be used to iterate over any collection, whether it is an object or an array. In the case of an array, the callback is passed an array index and a corresponding array value each time. (The value can also be accessed through the `this` keyword, but Javascript will always wrap the `this` value as an `Object` even if it is a simple string or number value.) The method returns its first argument, the object that was iterated.

- bike02.js

```
$(function(){
    parseBike();
});

function parseBike(){
    $.getJSON("json/bike.json", function(data){
        $.ajax({
            url : "bike.do",
            method : "POST",
            data : {"obj":JSON.stringify(data),"command":"second_db"}, // servlet으로 보낼때는 data 지정, 받을때는 dataType 지정
            success : function(msg) {
                if(msg > 0) {
                    alert(msg);
                    $.each(data, function(key, val){
                        if (key == "DESCRIPTION") {
                            $("table").attr("border",1);
                            var $tr = $("<tr>"); // createElement
                            for(var i in val) {
                                $tr.append($("<th>").html(val[i]));
                                // for(var i in val)의 성질
                                // i 찍으면 key가 나오고
                                // val[i]를 찍으면 value가 나온다
                            }
                            $("thead").append($tr);
                        } else {
                            for (var i = 0; i < val.length; i++) {
```

```

        var $tr = $("<tr>");
        for(var j in val[i]) { // DATA 는 현재 배열형태로 저장되어있음. i 는 0번지 j는 value임 그러니까 0번지의 0번째면 서울특별시
            $tr.append($("<td>").html(val[i][j]));
        }
        $("tbody").append($tr);
    }
    });
} else {
    alert("db 저장 실패...");
}
},
error : function(){
    alert("통신 실패");
}
});
});
}
}

```

★ for(var i in val) { }

처음보는 for문 형태인데 {} 안에 i만 써서 출력하면 val의 속성(key)값이 나오고 val[i] 형태로 사용하게 되면 val의 value 값이 나오게 된다

해서 밑에 for(var j in val[i]){ }에 {}안에 var[i][j]라고하면 value값이 출력이 되고 j 라고 써버리면 마찬가지로 속성값만 출력이 된다.

• BikeController.java

```

package com.bike.controller;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.bike.dao.BikeDao;
import com.bike.dto.BikeDto;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

@WebServlet("/BikeController")
public class BikeController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doPost(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html; charset=utf-8");

        String command = request.getParameter("command");
        System.out.println "[" + command + ""];
        BikeDao dao = new BikeDao();

        if(command.equals("first")) {
            response.sendRedirect("bike01.jsp");
        } else if(command.equals("first_db")) {
            if(dao.delete()) {
                System.out.println("삭제 성공");
            }
        }

        String[] bikeList = request.getParameterValues("bike");
        List<BikeDto> list = new ArrayList<BikeDto>();

        for(int i=0; i<bikeList.length; i++) {
            String[] tmp = bikeList[i].split("/");
            BikeDto dto = new BikeDto(tmp[0],

```

```

        Integer.parseInt(tmp[1]),
        tmp[2],
        tmp[3],
        Integer.parseInt(tmp[4]),
        Double.parseDouble(tmp[5]),
        Double.parseDouble(tmp[6]));

    list.add(dto);
}

int res = dao.insertDb(list);
if(res > 0) {
    System.out.println("insert성공");
}else {
    System.out.println("insert실패");
}

response.sendRedirect("bike01.jsp");

} else if(command.equals("second")){
    response.sendRedirect("bike02.jsp");
}else if(command.equals("second_db")) {

    String objString = request.getParameter("obj");
    // System.out.println(objString);

    JsonElement element = JsonParser.parseString(objString);
    // JsonObject jsonObj = (JsonObject) element;

    //System.out.println(element.getAsJsonObject().get("DESCRIPTION"));
    // Object로 바뀌서 description 을 찾았음
    List<BikeDto> list = new ArrayList<BikeDto>();

    for(int i = 0; i<element.getAsJsonObject().get("DATA").getAsJsonArray().size(); i++) {
        JsonObject tmp = element.getAsJsonObject().get("DATA").getAsJsonArray().get(i).getAsJsonObject();
        // element 와 object의 차이점
        /*
        JsonObject : name,value 한쌍이다! {'String' , 'jsonelement'}
        JsonElement : Json요소임 -> JsonObject, JsonArray, JsonPrimitive(string,number,true,false), null
        --> value !!!!! 어떤 type이 올 지 모르기 때문에!

        object 안에는 element와 array가 들어가 있다.

        element를 JsonObject 형태로 바꿔주고 memberName인 "DATA"를 찾아(DATA는 배열형태인 문자열로 되어있음) JsonArray로 바꿔준 array사이즈를 찾은:
        element를 JsonObject 형태로 바꿔주고 memberName인 "DATA"를 찾아 JsonArray로 바꾸고 그 배열의 i번째를 다시 JsonObject형태로 바꿔준것,, 11

        */

        String addr_gu = tmp.get("addr_gu").getString();

        int content_id = tmp.get("content_id").getAsInt();

        JsonElement content_nm_je = tmp.get("content_nm");
        String content_nm = content_nm_je.getString();

        JsonElement new_addr_je = tmp.get("new_addr");
        String new_addr = new_addr_je.getString();

        int cradle_count = tmp.get("cradle_count").getAsInt();
        double longitude = tmp.get("longitude").getAsDouble();
        double latitude = tmp.get("latitude").getAsDouble();

        BikeDto dto = new BikeDto(addr_gu, content_id, content_nm, new_addr, cradle_count, longitude, latitude);
        list.add(dto);
    }

    if(dao.delete()) {
        System.out.println("삭제성공");
    }
    int res = dao.insertDb(list);
    if(res > 0) {
        System.out.println("저장 성공");
    }else {
        System.out.println("저장 실패");
    }

    response.getWriter().append(res+"");
}
}
}

```