

# Anyframe jQuery Plugin



Version 1.1.0

저작권 © 2007-2014 삼성SDS

본 문서의 저작권은 삼성SDS에 있으며 Anyframe 오픈소스 커뮤니티 활동의 목적하에서 자유로운 이용이 가능합니다. 본 문서를 복제, 배포할 경우에는 저작권자를 명시하여 주시기 바라며 본 문서를 변경하실 경우에는 원문과 변경된 내용을 표시하여 주시기 바랍니다. 원문과 변경된 문서에 대한 상업적 용도의 활용은 허용되지 않습니다. 본 문서에 오류가 있다고 판단될 경우 이슈로 등록해 주시면 적절한 조치를 취하도록 하겠습니다.

---

I. Introduction .....	1
II. jQuery .....	2
1. Superfish .....	3
1.1. Superfish 소개 .....	3
1.2. Superfish 적용하기 .....	3
2. jqGrid .....	5
2.1. jqGrid를 위한 Spring MVC Controller 구현 .....	5
2.1.1. MovieSearchVO 생성 .....	5
2.1.2. MovieFinderController.list() .....	5
2.2. jqGrid를 이용한 HTML 개발 .....	6
2.2.1. jqGrid를 위한 Javascript 라이브러리 dependency .....	6
2.2.2. jqGrid Movie Finder .....	6
3. Quickpager .....	9
3.1. jqgrid와 quickpager 연동 .....	9
4. jstree .....	10
4.1. jsTree의 활용 .....	10
5. Upload .....	17
5.1. uploadify 소개 .....	17
5.2. jqueryUpload.js .....	17
6. jquery-ui .....	21
6.1. Autocomplete .....	21
6.2. Tab widget .....	22
6.3. Dialog widget .....	24
6.4. Button widget .....	25
7. Validation .....	26
III. jQuery UI sample .....	28
8. Grid(jqGrid) .....	29
8.1. Grid 화면 소개 .....	29
8.2. Grid 화면 .....	29
9. Grid(jqGrid) + Form .....	30
9.1. Grid + Form 화면 소개 .....	30
9.2. Grid + Form 화면 .....	30
10. Tree Grid(jqGrid) .....	32
10.1. Tree Grid 화면 소개 .....	32
10.2. Tree Grid 화면 .....	32
11. Tree Grid(jqGrid) + Form .....	34
11.1. Tree Grid + Form 화면 소개 .....	34
11.2. Tree Grid + Form 화면 .....	34
12. 기타 .....	36
12.1. 참고 사이트 .....	36

---

# I.Introduction

jQuery Plugin은 AJAX를 활용한 Spring MVC 기반의 웹 어플리케이션 개발 사례를 제공할 목적으로 만들어졌다. jQuery Plugin은 JSON 형태의 데이터를 이용한 공통 Controller 클래스 및 jQuery Javascript 프레임워크 및 이를 바탕으로 개발된 오픈소스 UI Component를 활용하여 작성된 샘플 코드와 이를 활용하는데 필요한 참조 라이브러리들로 구성되어 있으며, Plugin 샘플 코드에 활용된 jQuery UI 컴포넌트들은 jqgrid, quickpager, jstree, jquery-ui(button, dialog, tab, autocomplete), uploadify, superfish 등이 있다.

## Installation

Command 창에서 다음과 같이 명령어를 입력하여 jquery plugin을 설치한다.

```
mvn anyframe:install -Dname=jquery
```

installed(mvn anyframe:installed) 혹은 jetty:run(mvn clean jetty:run) command를 이용하여 설치 결과를 확인해볼 수 있다.

## Dependent Plugins

Plugin Name	Version Range
Query [ <a href="http://dev.anyframejava.org/docs/anyframe/plugin/optional/query/1.6.0/reference/htmlsingle/query.html">http://dev.anyframejava.org/docs/anyframe/plugin/optional/query/1.6.0/reference/htmlsingle/query.html</a> ]	2.0.0 > * > 1.4.0

---

## II.jQuery

---

---

# 1. Superfish

Jquery Plugin은 화면의 메뉴 구성을 위해 Superfish 라이브러리를 사용한다.([http://users.tpg.com.au/j\\_birch/plugins/superfish/](http://users.tpg.com.au/j_birch/plugins/superfish/))

## 1.1. Superfish 소개

Superfish는 jQuery를 기반으로하는 자바스크립트 라이브러리로서 CSS를 이용한 Drop-down 메뉴를 제공한다.

Superfish가 제공하는 기능은 메뉴 애니메이션, 탭키를 이용한 메뉴 이동, 다양한 Callback 함수 지원 등이 있다.

## 1.2. Superfish 적용하기

Superfish를 적용하기 위한 순서는 다음과 같다.

- superfish.js와 적용할 CSS 파일을 다음과 같이 추가한다. 기본적인 CSS 파일은 Superfish 라이브러리가 제공한다.

```
<script type="text/javascript" src="<c:url value='/jquery/jquery/superfish/superfish.js'/>"></script>
<link rel="stylesheet" media="screen" href="<c:url value='/jquery/jquery/superfish/superfish.css'/>" />
<link rel="stylesheet" media="screen" href="<c:url value='/jquery/jquery/superfish/superfish-vertical.css'/>" />
```

- CSS의 ul Element에 대해서 superfish() 함수를 호출하는 자바 스크립트를 다음처럼 추가해준다.

```
$(document).ready(function() {
    $('ul.sf-menu').superfish();
});
```

- 다음은 jquery Plugin에서 작성한 JSP 코드이다. ul.sf-menu 속성의 css가 적용된 태그에 Superfish를 적용한다.

```
<ul class="sf-menu sf-vertical">
  <li>
    <a href="<c:url value='/jqueryMovieFinder.do?method=listView'/>">Grid Example</a>
  </li>

  <li>
    <a href="<c:url value='/jqueryMovieTree.do?method=treeView'/>">Tree Example</a>
  </li>
</ul>
```

위와 같이 정의한 Superfish는 아래와 같은 메뉴를 출력하게 된다



---

Grid Example

---

Tree Example

---

Copyright 2012 [www.anyframejava.org](http://www.anyframejava.org)

---

## 2.jqGrid

jqGrid는 웹 상에서 tabular data를 표현하고 조작하기 위한 솔루션을 제공하는 AJAX 기반 자바스크립트 컨트롤러이다.(<http://www.trirand.com/jqgridwiki/doku.php>)

### 2.1.jqGrid를 위한 Spring MVC Controller 구현

#### 2.1.1.MovieSearchVO 생성

MovieSearchVO는 jqGrid에서 parameter로 제공하는 page, sord, sidx를 멤버로 가지고 있는 클래스이다.

MovieSearchVO는 SearchVO 클래스를 상속받는데, SearchVO 클래스에는 영화 검색시 키워드를 저장할 멤버 변수를 갖고 있다.

다음은 jqGrid 적용을 위해서 작성한 MovieSearchVO의 코드 일부이다.

```
package org.anyframe.plugin.jquery.domain;

public class MovieSearchVO extends SearchVO{

    private String nowPlayingCondition = "Y";

    private String sord;

    private String sidx;

    private int page = 1;

    ... 중략 ...
```

#### 2.1.2.MovieFinderController.list()

현재 jQuery plugin에서는 jQuery의 Grid 컴포넌트(jqgrid plugin)에서 Grid를 그릴 때 비즈니스 서비스 호출 후 반환되는 Page 객체를 바로 받을 수 있는 것이 아닌 Grid에서 인식할 수 있는 Key와 Value 쌍의 Map 형태로 Model 객체에 셋팅해줘야 한다. 다음은 MovieFinderController 클래스의 일부이다.

```
@RequestMapping(params = "method=list")
public String list(MovieSearchVO search, Model model) throws Exception {

    Page resultPage = movieFinder.getPagingList(search);

    model.addAttribute("page", String.valueOf(resultPage.getCurrentPage()));
    model.addAttribute("total", String.valueOf(resultPage.getMaxPage()));
    model.addAttribute("records", String.valueOf(resultPage.getTotalCount()));
    model.addAttribute("rows", resultPage.getList());
    return "jsonView";
}
```

위의 코드에서 볼 수 있듯이 비즈니스 서비스 수행후 Return 값이 org.anyframe.pagination.Page 타입일 경우 jQuery의 Grid에서 인식 할 수 있는 key값으로 jsonModel 객체를 셋팅해 주고있다.

## 2.2.jqGrid를 이용한 HTML 개발

### 2.2.1.jqGrid를 위한 Javascript 라이브러리 dependency

jqGrid를 사용하기 위해서는 jquery, jquery-ui, jqGrid 라이브러리가 필요하다.

```
<!-- jquery -->
<script type="text/javascript" src="<c:url value='/jquery/jquery/jquery-1.7.2.min.js' />"></script>
<script type="text/javascript" src="<c:url value='/jquery/jquery/superfish/superfish.js' />"></script>

<!-- jquery ui-->
<script type="text/javascript" src="<c:url value='/jquery/jquery/jquery-ui/jquery-ui-1.8.22.custom.min.js' />"></script>
<link href="<c:url value='/jquery/jquery/jquery-ui/themes/smoothness/jquery-ui-1.8.22.custom.css' />" rel="stylesheet" type="text/css" />

<!-- jqGrid -->
<script type="text/javascript" src="<c:url value='/jquery/jquery/jqgrid/i18n/grid.locale-en.js' />"></script>
<script type="text/javascript" src="<c:url value='/jquery/jquery/jqgrid/jquery.jqGrid.min.js' />"></script>
<link href="<c:url value='/jquery/jquery/jqgrid/ui.jqgrid.css' />" rel="stylesheet" type="text/css" />
```



#### 참고

※ 라이브러리 참조 선언의 순서에 주의하여야 한다. jQuery -> jQuery ui -> jqGrid 순으로 정의를 한다.

※※ jqGrid의 경우 필요에 따라 추가적인 플러그인 라이브러리를 호출하여 사용할 수 있다.(jqGrid 위키사이트 참조 : <http://www.trirand.com/jqgridwiki/doku.php>)

다음은 jquery plugin 에서는 제공하는 jqGrid MovieFinder 샘플이다.

### 2.2.2.jqGrid Movie Finder

- i. : 영화 리스트를 제공
- ii. : row 단위로 select 할 수 있으며, quickpager 오픈소스 컴포넌트와 결합하여 page navigation을 제공
- iii. : jquery-ui에서 제공하는 dialog widget과 연동하여 영화 등록/수정 기능을 제공
- iv. : jquery-ui에서 제공하는 autocomplete 와 연동하여 목록에 대한 검색기능을 제공

다음은 영화 목록을 출력하는 list.jsp파일의 일부이다.

```
jQuery(document).ready( function() {
    jQuery('#grid').jqGrid(
    {
        url: "<c:url value='/jqueryMovieFinder.do?method=list' />",
        mtype: 'POST',
```



```

datatype : "json",
colNames : [ '<spring:message code="movie.genre" />', 'id', '<spring:message
code="movie.title" />', '<spring:message code="movie.director" />'
, '<spring:message code="movie.actors" />', '<spring:message code="movie.releaseDate" /
>'],
jsonReader: {
    repeatitems: false
},
colModel : [
    {name : 'genre.name', index : 'genre.name', align : 'center'},
    {key : true, name : 'movieId', hidden : true},
    {name : 'title', index : 'title', align : 'center'},
    {name : 'director', index : 'director' , align : 'center'},
    {name : 'actors', index : 'actors' , align : 'center'},
    {name : 'releaseDate', index : 'releaseDate', align : 'center'}],
autowidth : true,
height : 69,
scroll : false,
//forceFit:true,
multiselect : true,
viewrecords : true,
rowNum : 3,
sortable : true,
loadComplete : function(xhr) {
    $('#pagination').quickPager( {
        pageSize: '3',
        pageUnit: '3',
        pageIndexId: 'pageIndex',
        searchButtonId: 'searchMovies',
        currentPage: jQuery('#grid').getGridParam('page'),
        totalCount: jQuery('#grid').getGridParam('records'),
        searchUrl: '#'
    });
},
gridComplete : function() {
},
loadError: function(xhr,st,err) {
    alert('<spring:message code="error.moviefinderimpl.getpaginglist" />');
},
ondblClickRow: function(rowId) {
    var rowData = jQuery('#grid').getRowData(rowId);

    openMovieFormDialog('edit', rowData.movieId);
},
beforeSelectRow: function (rowid, e) {
    //Default multi selection works whenever a row of jqGrid clicked.
    //Configuring beforeSelectRow for Multi selection to work when check boxes are clicked
- not the rows.
    var $myGrid = $(this),
        i = $.jgrid.getCellIndex($(e.target).closest('td')[0]),
        cm = $myGrid.jqGrid('getGridParam', 'colModel');
    return (cm[i].name === 'cb');
}
});
.
.
.
중략

```

위와 같이 jqgrid로 구현된 리스트의 모습은 아래와 같다.

## • Search List of Movie

Title:  Now Playing: 

<input type="checkbox"/>	Genre	Title	Director	Actors	
<input type="checkbox"/>	Adventure	Alice in Wonderland	Tim Burton	Johnny Depp	
<input type="checkbox"/>	Sci-Fi	Avatar	James Cameron	Sigourney Weaver	
<input type="checkbox"/>	Action	Green Zone	Paul Greengrass	Yigal Naor	

| 1 | 2 | Next ▶

## 3.Quickpager

jqgrid는 기본적으로 Paging 처리를 위한 Pager를 제공하고 있다. Anyframe에서는 pagenavigator와 유사한 Pager 출력을 위해 quickpager를 확장하여 사용하고 있다. quickpager를 사용하기 위해서는 리스트 Script내의 loadComplete 함수 안에 paging 정보를 셋팅 해주고 search 버튼을 클릭하는 이벤트를 발생 시키도록 한다.

### 3.1.jqgrid와 quickpager 연동

관련 jQuery 코드는 다음과 같다.

```
jQuery("#grid").jqGrid({
...중략...
loadComplete : function(xhr) {
    $(paginationId).quickPager( {
        pageSize: "10",
        pageUnit: "10",
        pageIndexId: 'grid_' + id + "_pageIndex",
        searchButtonId: 'grid_' + id + "_btnSearch",
        currentPage: $(gridId).getGridParam("page"),
        totalCount: $(gridId).getGridParam("records"),
        searchUrl: "#"
    });
},
...중략...
});

$(gridId + "_btnSearch").click(function() {
    $(gridId).setGridParam({
        page: $(gridId + "_pageIndex").val(),
        postData: {
            searchKeyword:$(gridId + "_searchKeyword").val(),
            searchCondition:$(gridId + "_searchCondition").val()
        }
    });
    $(gridId).setGridParam({url:"<c:url value='/jqueryBoard.do?method=list'/>"})
    >").trigger("reloadGrid");
});
```

위와 같이 Script 코드가 작성 되면 pagenavigator 출력 부분에 아래와 같이 div 영역을 표시해준다.

```
<div id="${gridId}_boardNav">
  <div id="${gridId}_pagination" class="pagination"></div>
</div>
```

위와 같이 정의한 quickpager는 아래와 같은 pagenavigator를 출력하게 된다



#### jqgrid에서 제공하는 pager

jqgrid에서도 paging 처리를 위한 간편한 pager를 제공한다. 구현 방법은 아래와 같다.

```
//jqgrid 속성 설정 내에 정의
pager : jQuery('#pager')

<!-- JSP 내의 pager 출력 부분에 정의 -->
<div id="pager" class="scroll" style="text-align: center;"></div>
```

---

## 4.jstree

jstree는 계층적으로 조직된 데이터를 tree 형태로 보여주기 위해 제공되는 jQuery 기반의 오픈소스 UI 컴포넌트이다.

jsTree is a javascript based, cross browser tree component. It is packaged as a jQuery plugin (<http://www.jstree.com>)

### 4.1.jsTree의 활용

jstree는 0.9.9a 이후 1.0-rc3로 업그레이드 되면서 일부 api 및 사용법이 변경되었다. 여기서는 1.0-rc3 버전을 기준으로 설명한다.

jstree는 크게 html, json, xml 방식의 data loading 방식을 제공하며, 여기서는 html 방식을 제공한다. html은 JSTL을 사용하여 다음과 같이 표시 할 수 있다.

```
<!-- start of tree -->
<div id="tree">
  <span>listNode</span>
  <ul>
    <li id="ROOT" rel="root">
      <a href='#>ROOT</a>
      <c:set var="prevDepth" value="-1"/>
      <c:forEach var="node" items="${treeList}">
        <c:if test="${node.depth > prevDepth}">
          <ul>
        </c:if>
        <c:if test="${prevDepth > node.depth}">
          <c:forEach begin="${node.depth}" end="${prevDepth - 1}" step="1">
            </ul></li>
          </c:forEach>
        </c:if>
        <li id="${node.nodeId}" parentId="${node.parentId}" depth="${node.depth}"
          rel="${node.type}">
          <a href='#>${node.nodeName}</a>
          <c:if test="${node.hasChild == 0}">
            </li>
          </c:if>
          <c:set var="prevDepth" value="${node.depth}"/>
        </c:forEach>
      </li>
    </ul>
  </div>
<!-- end of tree -->
```

jsTree를 사용하여 트리를 구성하면 다음과 같다.

```
// tree definition
$(document).ready(function() {

  $("#tree").jstree({
    'plugins' : ["themes", "json_data", "ui", "types", "contextmenu", "crrm"], //, "dnd"
    "html_data" , 'checkbox', "cookies",
    'themes' : {
      'theme' : 'apple',
      'dots' : false,
      'icons' : true,
      'url' : "jquery/jquery/jstree/themes/apple/style.css"
```

```

    },
    'json_data' : {
        'ajax':{ //# Tree data 가져옴.
            "url" : "<c:url value='/jqueryMovieTree.do?method=tree' />",
            "data" : function(n){
                var return_id = $('#searchKeyword').val();
                if(n.attr) {
                    var id = n.attr("id");
                    var pid = n.parents('li:eq(0)').attr('id');
                    if(pid == undefined) {
                        return {
                            searchKeyword : id,
                            id : id
                        };
                    } else {
                        return {
                            id:id,
                            searchKeyword : return_id;
                        }
                    } else {
                        return {
                            id : 0,
                            searchKeyword : return_id;
                        }
                    }
                },
            },
            "success" : function(data){
                return data.JSTreeNodeList;
            }
        }
    },
    'types' : {
        'types' : {
            'root' : {
                'icon' : {'image' : '<c:url value="/jquery/image/icons.png"/>'}
            },
            'lockedroot' : {
                'icon' : {'image' : '<c:url value="/jquery/image/lockedicons.png"/>'}
            },
            'leaf' : {
                'icon' : {'image' : '<c:url value="/jquery/image/leaficons.png"/>'}
            }
        }
    },
    'contextmenu' : {
        'items' : createContextMenu
    }
}).bind("remove.jstree", function(e, data) { // event handling for node delete
    if(confirm("Are you sure you want to delete this movie?")){
        data.rslt.obj.each(function() {
            $.ajax({
                async : false,
                type : 'POST',
                url : "<c:url value='/jqueryMovieTree.do?method=remove' />",
                data : {
                    "movieId" : $(data.rslt.obj).attr("id")
                },
                error : function() {
                    $.jstree.rollback(data.rlbk);
                }
            });
        });
    }
});

```

```

}else{
    $.jstree.rollback(data.rlbk);
}
}).bind("select_node.jstree", function (e, data) { // event handling for node select
    var node = data.rslt.obj;
    var id = node.attr('id');
    var pid = node.parents('li:eq(0)').attr('id');

    if(pid == undefined){//genre

        $("#tabs").show();
        $("#tabs").tabs("enable", 0);
        $("#tabs").tabs("select", 0);
        $("#tabs").tabs("disable", 1);

        //GET Genre
        $.getJSON("<c:url value='jqueryMovieTree.do?method=getGenre'/>&genreId=" + id,
function(data){
    document.genreForm.genreId1.value = data.genre.genreId;
    document.genreForm.name.value = data.genre.name;
});
}else{//movie

        $("#tabs").show();
        $("#tabs").tabs("enable", 1);
        $("#tabs").tabs("select", 1);
        $("#tabs").tabs("enable", 0);

        $("#releaseDate").datepicker({dateFormat: "yy-mm-dd", autoSize:true});

        $.ajaxSetup({
            "error":function() {
                alert('<spring:message code="error.movieserviceimpl.get" />');
                $("#tabs").hide();
            }
        });

        //GET Movie
        $.getJSON("<c:url value='/jqueryMovieTree.do?method=get' />&movieId=" + id,
function(data){
    document.movieForm.movieId.value = data.movie.movieId;
    document.movieForm.title.value = data.movie.title;
    document.movieForm.director.value = data.movie.director;
    document.movieForm.actors.value = data.movie.actors;
    document.movieForm.genreId.value = data.movie.genre.genreId;

    if(data.movie.nowPlaying == "Y"){
        document.movieForm.nowPlaying.checked = true;
    }
    else{
        document.movieForm.nowPlaying.checked = false;
    }

    //for safari
    document.getElementById("releaseDate").value=data.movie.releaseDate;

    if(document.getElementById("releaseDate").value=="null"){
        document.getElementById("releaseDate").value="";
    }
    document.movieForm.runtime.value = data.movie.runtime;

    document.movieForm.filePaths.value = data.movie.filePaths;
    if(data.movie.filePaths == null || data.movie.filePaths == ""){

```

```

        $("#imgSrc").hide();
        $("#imgTxt").show();
    }
    else{
        jQuery("#poster").attr("src", "${ctx}"+document.movieForm.filePaths.value);
        $("#imgSrc").show();
        $("#imgTxt").hide();
    }

    document.movieForm.ticketPrice.value = data.movie.ticketPrice;

    $('#genreId').msDropDown();

    });

    $.getJSON("<url value='/jqueryMovieTree.do?method=getGenre' />&genreId=" + pid,
function(data){
    document.genreForm.genreId1.value = data.genre.genreId;
    document.genreForm.name.value = data.genre.name;
    });
}
$("#tabs").show();
});

});

```

트리의 노드 타입에 따른 컨텍스트 메뉴 구성은 다음과 같이 구현한다.

```

/**
 * context menu generate for tree
 */
function createContextMenu(node) {
    var default_object = {
        remove : {}
    };

    var pid = node.parents('li:eq(0)').attr('id');
    var id = node.attr('id');
    if(pid == undefined){//genre
        default_object = {
            remove : {
                label : '<spring:message code="movie.button.remove" />',
                _disabled : true
            }
        };
    }
    }else{//movie
        default_object = {
            remove : {
                label : '<spring:message code="movie.button.remove" />',
                action : function(obj) {
                    if(this.is_selected(obj)) {
                        this.remove();
                    } else {
                        this.remove(obj);
                    }
                }
            }
        };
    }
    return default_object;
}

```

다음은 Tree에 표시할 데이터를 만드는 Controller 클래스이다. Tree의 상위 노드는 Genre가 되고 하위 노드는 영화 Title이 된다.

```
@Controller("jqueryMovieTreeController")
@RequestMapping("/jqueryMovieTree.do")
public class MovieTreeController {

    ...중략...

    @RequestMapping(params = "method=tree")
    public String tree(@RequestParam("id") String id, MovieSearchVO search, Model model) throws
    Exception {
        List<JSTreeNode> nodeList = buildTree(id,search);
        model.addAttribute("JSTreeNodeList", nodeList);
        return "jsonView";
    }

    private List<JSTreeNode> buildTree(String id, MovieSearchVO search) throws Exception {
        JSTreeNode node = null;
        Attributes attribute = null;
        List<?> jsTreeList = null;
        List<JSTreeNode> nodeList = new ArrayList<JSTreeNode>();

        if (id.equals("0")) {
            //If id equals to "0", it means jsTree need a list of genre to generate its root. (root
            is a genre in this sample)
            jsTreeList = genreService.getGenreList(search);
            for (int i = 0; i < jsTreeList.size(); i++) {
                Genre genre = (Genre) jsTreeList.get(i);
                node = new JSTreeNode();
                attribute = new Attributes();

                attribute.setId(genre.getGenreId());

                node.setAttr(attribute);
                node.setData(genre.getName());
                //if the value of state is "closed", jsTree generates an expendable button on the left
                of its folder image.
                node.setState(genre.getState());
                if (genre.getState() == null || "".equals(genre.getState()))
                    attribute.setRel("lockedroot");
                else
                    attribute.setRel("root");
                nodeList.add(node);
            }
        } else {
            /*
             * If id does not equal to "0", it means jsTree need a leaf data of the selected root.
             (In this case, id is one of genres id, not "0")
             * The leaf data is a list of movie.
             */
            jsTreeList = movieFinder.getListByCategory(search);
            if (jsTreeList.size() > 0) {
                for (int i = 0; i < jsTreeList.size(); i++) {
                    Movie movie = (Movie) jsTreeList.get(i);

                    node = new JSTreeNode();
                    attribute = new Attributes();

                    attribute.setId(movie.getMovieId());
```



```
        attribute.setRel("leaf");
        node.setAttr(attribute);
        node.setData(movie.getTitle());
        nodeList.add(node);
    }
}
return nodeList;
}

...종략...
}
```

위의 예제에서 buildTree() 매소드로 jsTree에 출력할 Node List를 생성한 후, 결과값을 Model에 저장한다. 이 때 Key 값을 'JSTreeNodeList'로 설정하였다.

'JSTreeNodeList'는 jsTree가 Controller로 부터 넘어온 JSON 객체에서 바인딩할 데이터를 가져올 때 Key 값으로 사용한다.

다음은 jstree를 이용하여 Tree를 출력한 화면이다.

sample

sample

## • Search List of Movie

Genre:



- ▼ Action
  - Green Zone
- ▼ Adventure
  - Alice in Wonderland
- Animation
- ▼ Comedy
  - She is Out of My League
- ▼ Crime
  - Shutter Island
- ▼ Drama
  - Remember Me
- Fantasy
- Romance
- ▼ Sci-Fi
  - Avatar
- Thriller

---

## 5.Upload

jQuery와 AJAX를 활용한 Multi file 첨부기능을 구현하여 제공하고 있다.

### 5.1.uploadify 소개

uploadify는 jquery와 flash Object를 통하여 간편하게 file 첨부을 구현할 수 있게 해주는 오픈소스 컴포넌트이다. 자세한 내용은 <http://www.uploadify.com/> 사이트를 참조하기 바란다.

### 5.2.jqueryUpload.js

jquery plugin에서는 uploadify를 사용하여 파일첨부를 구현한 별도의 서브셋을 jqueryUpload.js 에 별도로 구현하였다. 이를 통해서 파일 첨부 관련 코드가 비즈니스 로직에 추가되는 것을 최소화하도록 의도된 것이다.

파일 업로드 컴포넌트는 다음과 같이 인스턴스화 시킨다.

```
$(document).ready(function() {  
    function drawUploadPane() {  
        $('#uploadPane').attachment({  
            'contextRoot' : '${ctx}',  
            'callBack' : function() {  
                postMovieForm();  
            }  
        });  
    }  
});  
}
```

위의 코드에서 'uploadPane', 즉, 첨부파일 UI가 표시될 영역은 영화 등록 form 인 'dialog' 영역에 선언되어 있다.

아래는 영화 등록 form 이다.

```
<!-- Movie Form start -->  
<div id="dialog" class="dialog">  
<form:form modelAttribute="movie" method="post" id="movieForm" name="movieForm">  
<table width="400px">  
    <colgroup>  
        <col style="width:35%;" />  
        <col style="width:65%;" />  
    </colgroup>  
<tbody>  
    ...중략...  
    <th><label for="posterFile"><spring:message code="movie.posterFile"/>&nbsp;</label></th>  
    <td>  
        <div id="imgPane">  
            <img id="poster" src="" alt="<spring:message code='movie.posterFile'/">" border="0"  
width="80" height="100" />  
        </div>  
        <div id="uploadPane"></div>  
    </td>  
</tr>  
</tbody>  
</table>  
</form:form>  
</div>
```

영화 등록 form은 업로드한 이미지가 없는 경우 업로드 UI를 보여주고, 이미지가 있는 경우 미리보기 화면을 제공한다. 이는 jquery를 이용해서 구현한다.

```
$('#filePaths').val(data.movie.filePaths);
if(data.movie.filePaths != "") {
    $('#poster').attr("src", '{ctx}'+data.movie.filePaths);
    $('#imgPane').show();
} else {
    $('#uploadPane').show();
}
```

다음은 upload 경로 등의 정보를 담고 있는 jqueryUpload.js의 일부분이다.

```
$("#uploadify").uploadify({
    swf : options.contextRoot + '/jquery/jquery/uploadify/uploadify.swf' ,
    uploader : options.contextRoot + '/jqueryUploadFile.do',
    queueID : "fileQueue",
    fileObjName : "fileData",
    auto : false,
    multi : false,
    width : 80,
    height : 24,
    debug : false,
    removeTimeout : 0,
    fileTypeExts : '*.jpg; *.gif; *.png;',
    fileSizeLimit : 10000000,
    buttonImage : options.contextRoot + '/jquery/image/uploadBrowse.png',
    onUploadSuccess : function(file, data, response) {
        .
        .
        .
        중략
    }
})
```

파일을 업로드한 후 실제 게시물을 등록시켜야 하므로 실제로 게시물을 저장하는 스크립트 함수인 postMovieForm()을 callback으로 선언한다.

```
function postMovieForm(){
    if(dialogMode == 'add') {
        $.post('<url value="/jqueryMovie.do?method=create"/>',
            $('#movieForm').serialize(),
            function(data) {
                reloadGrid();
                $('#movieForm').dialog('close');
            });
    } else if(dialogMode == 'edit') {
        $.post('<url value="/jqueryMovie.do?method=update"/>', $('#movieForm').serialize(),
            function(data) {
                reloadGrid();
                $('#movieForm').dialog('close');
            });
    } else {
        logger.log('dialogMode is invalid : ' + dialogMode);
    }
}
```

파일 업로드를 구현한 UploadController.java는 다음과 같다.

```
@Controller("jqueryUploadController")
@RequestMapping("/jqueryUploadFile.do")
public class UploadController {
    private String uploadPath = "/upload";

    @RequestMapping
    public String uploadFile(@RequestParam(value = "fileData", required = false) MultipartFile
        file,
        Model model, HttpServletRequest request) throws Exception {

        String destDir = request.getSession().getServletContext().getRealPath(
            uploadPath);

        File repositoryDir = new File(destDir);
        if (!repositoryDir.exists()) {
            boolean created = repositoryDir.mkdirs();
            if (!created) {
                throw new Exception(
                    "Fail to create a directory for attached file ["
                        + repositoryDir + "]);
            }
        }

        SimpleDateFormat formatter = new SimpleDateFormat("yyyyMMddHHmmssSSS",
            new Locale("ko", "KR"));

        String realFileName = file.getOriginalFilename();
        String fileNameExtension =
            realFileName.substring(realFileName.lastIndexOf(".")).toLowerCase();
        String fileId = "FILE-" + formatter.format(new Date());
        String convertedFileName = fileId + fileNameExtension;
        String filePathToBeStored = uploadPath + "/" + convertedFileName;

        file.transferTo(new File(destDir + "/" + convertedFileName));

        model.addAttribute("filePaths", filePathToBeStored);
        model.addAttribute("realFileName", realFileName);

        return "jsonView";
    }
}
```



## Weblogic을 WAS로 사용하는 경우 주의사항

1. Flash 동작오류 : uploadify는 Upload 기능을 Flash를 이용하여 구현한다. Weblogic과 IE를 사용하는 경우 Flash가 정상적으로 동작하지 않을 수 있다. 이런 경우 web.xml 파일에 'mime-mapping'을 추가한다.

```
<mime-mapping>
  <extension>swf</extension>
  <mime-type>application/x-shockwave-flash</mime-type>
</mime-mapping>
```

2. WAR 로 Weblogic에 배포하는 경우 오류 : War로 배포하는 경우, Upload Path를 못 가져올 수 있다. 이런 경우 다음과 같이 조치한다.

- Weblogic Console 화면에서 Domain -> Web Application -> Archive Path 체크 박스 선택



## 참고

※ jqueryUpload.js 내에 구현된 내용은 하나의 구현 사례이므로 구현 요건에 따라 자유롭게 재구성될 수 있다.

다음은 uploadify와 jqueryUpload.js를 활용하여 파일첨부 기능을 구현한 것이다.

Movie Information

Title	<input type="text"/>
Director	<input type="text"/>
Genre	Action <input type="button" value="v"/>
Actors	<input type="text"/>
Runtime	<input type="text"/> min.
Release Date	<input type="text"/>
Ticket Price	<input type="text"/>
Now Playing	Is this movie now playing ? <input checked="" type="checkbox"/>
Poster	<div>jquery_upload.png (11KB)</div> <div><input checked="" type="checkbox"/> Browse</div>

Save

Cancel

---

## 6.jquery-ui

jQuery는 UI 플러그 인을 통해 UI컴포넌트를 추가적으로 제공하며, 테마 기능과 연동된 UI컴포넌트를 통해 강력한 User Interface를 Pure-HTML 환경에서도 구현할 수 있도록 도와주고 있다.

jQuery 에서 제공하는 UI 컴포넌트에 대한 자세한 기능은 <http://jqueryui.com> 에서 확인하기 바란다.

Anyframe jQuery plugin에서는 jQuery ui 1.8.16 버전을 바탕으로 autocomplete, tab, dialog, button, theme 기능을 활용하여 제공하고 있다.

### 6.1.Autocomplete

autocomplete는 사용자가 입력한 prefix를 가지고 자동 완성 기능을 제공하는 UI컴포넌트이다.

아래 자바스크립 코드는 게시물 리스트의 검색에 autocomplete 기능을 적용한 내용이다. success 부분과 select 부분의 코드활용을 주목하기 바란다.

```
$("#searchKeyword").autocomplete({
    source: function( request, response ) {
        $.ajax({
            type : 'POST',
            url : "<c:url value='/jqueryMovie.do?method=getMovieTitleList' />",
            contentType : 'application/x-www-form-urlencoded;charset=UTF-8',
            //data : jsonData,
            data: {
                term: request.term
            },
            dataType: 'json',
            success : function(data){
                response($.map(data.autoData, function(item) {
                    return {
                        label: item.title,
                        value: item.title
                    }
                }));
            },
            error : function(data) {
                alert("[autoComplete error] Sending data to designated url is not working. Data : " + data);
            }
        });
    }
});
```

다음은 Autocomplete을 적용한 결과이다.

## • Search List of Movie

Title: A

Now Playing: Playing

<input type="checkbox"/>	Genre	Title	Director	Stars	Release Date
<input type="checkbox"/>	Adventure	Alice in Wonderland	Tim Burton	Liam Neeson, Johnny Depp	2010-03-04
<input type="checkbox"/>	Sci-Fi	Avatar	James Cameron	Sigourney Weaver	2010-12-16
<input type="checkbox"/>	Action	Green Zone	Paul Greengrass	Yigal Naor	2010-03-25

1 | 2 | 3 | Next ▶



Add



Delete

## 6.2.Tab widget

Tab 위젯은 동일한 목적을 가지고 있으나 성격이 상이한 UI를 분할하여 화면 복잡도를 낮추고 효율적인 User Interface 구현을 도와주는 자바스크립트 기반의 UI 컴포넌트이다. (<http://api.jqueryui.com/tabs>)

Tab 위젯을 사용하기 위해 jQuery UI를 다음과 같이 추가해 주었다.

```
<script type="text/javascript" src="<c:url value='/jquery/jquery/jquery-ui/jquery-ui-1.8.22.custom.min.js' />"></script>
<link href="<c:url value='/jquery/jquery/jquery-ui/themes/smoothness/jquery-ui-1.8.22.custom.css' />" rel="stylesheet" type="text/css" />
```

Jquery Plugin의 Tree 샘플에서는 두개의 탭을 사용한다. 다음은 Tab 위젯을 사용하기 위한 설정이다.

```
$("#tabs").tabs({remote : true, disabled: [0, 1]});
```

위의 코드에서 {remote : true} 옵션은 AJAX Loading 을 사용한다는 의미이고 {disabled: [0,1]} 옵션은 첫 번째, 두 번째 탭을 비활성화 한다는 것을 의미한다.

다음은 Tab 위젯을 구현한 jsTree 코드이며, 'enable' 과 'disable' 속성을 변경해서 탭 화면을 보여준다. Tree jsTree는 이곳을 참조한다.

```
...선택
.bind("select_node.jstree", function (e, data) { // event handling for node select
    var node = data.rslt.obj;
    var id = node.attr('id');
    var pid = node.parents('li:eq(0)').attr('id');

    if(pid == undefined){//genre

        $("#tabs").show();
        $("#tabs").tabs("enable", 0);
        $("#tabs").tabs("select", 0);
        $("#tabs").tabs("disable", 1);

        ...선택...

    });
}else{//movie
```



```

$("#tabs").show();
$("#tabs").tabs("enable", 1);
$("#tabs").tabs("select", 1);
$("#tabs").tabs("enable", 0);

... 생략 ...



});
}
$("#tabs").show();
});

```

왼쪽 트리에서 영화 선택 시 해당화면이 우측에 활성화 되며, 탭을 클릭해 탭 화면을 전환한다.



아래 그림은 Movie Information 탭을 클릭했을 때 이다.


#### • Search List of Movie

Genre:   

- ▶ Action
- ▼ Adventure
  - ☰ Alice in Wonderland
  - ☰ movie
- Animation
- ▶ Comedy
- ▶ Crime
- ▶ Drama
- Fantasy
- Romance
- ▶ Sci-Fi
- Thriller

Genre Information Movie Information













Title	<input type="text" value="Alice in Wonderland"/>
Director	<input type="text" value="Tim Burton"/>
Genre	Adventure 
Actors	<input type="text" value="Johnny Depp"/>
Runtime	<input type="text" value="110"/> min.
Release Date	<input type="text" value="2010-03-04"/>
Ticket Price	<input type="text" value="8000"/>
Now Playing	Is this movie now playing ? <input checked="" type="checkbox"/>
Poster	

 Update

아래 그림은 Genre Information 탭을 클릭했을 때 이다.

### • Search List of Movie

Genre:   

- ▶  Action
- ▼  Adventure
  -  Alice in Wonderland
  -  movie
-  Animation
- ▶  Comedy
- ▶  Crime
- ▶  Drama
-  Fantasy
-  Romance
- ▶  Sci-Fi
-  Thriller

Genre Information		Movie Information
ID	<input type="text" value="GR-02"/>	
Name	<input type="text" value="Adventure"/>	

## 6.3.Dialog widget

Dialog 위젯은 Window Popup이나 브라우저의 Message Alert Box를 Layed된 html 요소를 활용하여 대체할 수 있도록 한 것이다.


jQuery UI에서 제공하는 Dialog 위젯은 아래 코드와 같이 다양한 옵션 및 이벤트 핸들링을 구성할 수 있다.

```
$(document).ready(function() {
... 선택

// Dialog form definition for Category
$('#movieForm').dialog({
  autoOpen : false,
  width : "auto",
  height : "auto",
  maxWidth : "410px",
  title : "Movie Information",
  modal : true,
  resizable : false,
  close : function() {
    //close callback
  },
  open : function() {
    //open callback
  }
});
... 선택
```

다음은 Dialog를 적용한 결과이다. Modal 형태의 Window로서 'Cancel' 버튼과 타이틀바의 'x' 버튼 및 ESC 키를 통해 창을 닫을 수 있으며, window의 사이즈 조절도 가능하도록 옵션처리 되었다.

Movie Information ✕

Title	<input type="text" value="Alice in Wonderland"/>	
Director	<input type="text" value="Tim Burton"/>	
Genre	<input type="text" value="Adventure"/>	
Actors	<input type="text" value="Johnny Depp"/>	
Runtime	<input type="text" value="110"/>	min.
Release Date	<input type="text" value="2010-03-04"/>	
Ticket Price	<input type="text" value="8000"/>	
Now Playing	Is this movie now playing ? <input checked="" type="checkbox"/>	
Poster		

Save

Cancel

## 6.4.Button widget

jQuery UI에서 제공하는 버튼 위젯의 특징은 <button>태그를 그대로 활용한 다는 점으로 웹 표준을 그대로 준수하고 있다는 점이다.

```
<div class="buttons">
  <button id="${gridId}_btnAdd">Add</button>
  <button id="${gridId}_btnEdit">Edit</button>
  <button id="${gridId}_btnRemove">Remove</button>
  <button id="${gridId}_btnRefresh">Refresh</button>
</div>
```

위의 버튼 tag들에 다음과 같이 간단한 코딩으로 위젯을 적용하는 것이 가능하다. 이벤트 핸들링 또한 위젯적용여부에 관계없이 핸들링 하고 있다.

```
$("#button", ".buttons").button(); // 위젯 적용
// 'Add' 버튼의 onclick 이벤트 핸들링
$(gridId + "_btnAdd").click(function() {
  dialogMode = "add";
  AnyframeUpload.options.refId = '';
  $("#dialog-form").dialog( "open" );
});
```

## 7.Validation

jQuery에서는 validation plugin을 통해 jQuery를 적용한 AJAX 기반의 웹 어플리케이션의 form validation 수단을 제공한다.

validation plugin은 다양한 rule과 메시지 표시를 제공하지만, 예제에서는 가장 태그의 attribute를 통해 가장 간단히 구현하는 방법을 소개한다.

좀 더 자세한 내용은 <http://docs.jquery.com/Plugins/validation>을 참조하기 바란다.



### \* validation을 위한 css 설정

validation 메시지 표시를 위해 본 예제 코드에서는 jquery.css 에 다음의 내용을 추가하였다.

```
label.error {  
    float: none;  
    color: red;  
    padding-left: .5em;  
    display: block;  
}
```

validation 적용을 위해서는 해당되는form을 다음과 같이 지정한다.

```
$('#movieForm').validate();
```

해당 form의 html 은 아래와 같으며, class에 'required' 값을 부여함으로써 필수 필드로 지정된다. 또한 maxlength 값 지정을 통해 최대 길이를 제한할 수 있다.

```
<form:form modelAttribute="movie" method="post" id="movieForm" name="movieForm">  
  <table width="400px">  
    ... 생략...  
    <colgroup>  
      <col style="width:35%;" />  
      <col style="width:65%;" />  
    </colgroup>  
    <tbody>  
      <tr>  
        <th><label for="title"><spring:message code="movie.title"/>&nbsp;</label></th>  
        <td>  
          <form:input path="title" cssClass="required ct_input_g" cssStyle="width:150px;"  
size="40" maxlength="50" />  
        </td>  
      </tr>  
      <tr>  
        <th><label for="director"><spring:message code="movie.director"/>&nbsp;</label></th>  
        <td>  
          <form:input path="director" cssClass="required ct_input_g" cssStyle="width:150px;"  
size="40" maxlength="50" />  
        </td>  
      </tr>  
      ... 생략...  
    </tbody>  
  </table>  
</form:form>
```

실제 form 값들의 validation을 수행하기 위해서는 valid() 함수를 사용한다. 아래 그림에서 'save' 버튼을 클릭하면 saveMovie() 함수가 호출되고 영화 정보를 저장하기전 valid() 함수가 호출된다.

```
/**
function saveMovie() {
  if(!$('#movieForm').valid()) {
    logger.log('movieForm is invalid.');
```

... 생략

```
  return false;
}
}
```

다음은 validation을 게시물 등록/수정 form에 적용한 결과이다.

Movie Information

Title	<input type="text"/>	This field is required.
Director	<input type="text"/>	This field is required.
Genre	Action	
Actors	<input type="text"/>	This field is required.
Runtime	테스트	Please enter a valid number. min.
Release Date	2012-10-04	
Ticket Price	칠천원	Please enter a valid number.
Now Playing	Is this movie now playing ? <input checked="" type="checkbox"/>	
Poster	<input type="button" value="Browse"/>	

Save

Cancel

---

## III.jQuery UI sample

본 챕터에서는 총 4가지 케이스의 jQuery sample 화면을 제공한다. 각각의 화면은 커뮤니티의 카테고리, 커뮤니티, 각 커뮤니티의 게시글, 사용자 등을 관리하는 기능으로 구성되어 있다. 각 샘플 화면들은 UI 컴포넌트의 기본적인 사용법, 이벤트 핸들링, 스크립트 사용법 등을 보여준다.

## 8.Grid(jQgrid)

### 8.1.Grid 화면 소개

본 Grid 화면은 아래와 같은 UI 및 기능을 제공한다.

- UI : Grid(jqGrid), Grid Pagination, Button
- 기능 : 검색, 카테고리 목록 조회, 카테고리 추가/수정/삭제

### 8.2.Grid 화면

커뮤니티의 카테고리 목록을 조회하고 카테고리를 관리하는 기능을 갖추고 있는 화면이다. Grid 하단에 Pagination 기능을 제공하며 본 Sample에서는 해당 페이지 입력 혹은 클릭 시 해당 페이지의 데이터를 테이블에서 가져오는 방식으로 구현되어 있다.

CATEGORY GRID EXAMPLE

카테고리 이름		검색
이름	상세설명	등록일
컴퓨터	컴퓨터에 관련된 커뮤니티들이 모여있습니다.	2009-06-23
자동차	자동차에 관련된 커뮤니티들이 모여 있습니다.	2009-06-23
스포츠	스포츠에 관련된 커뮤니티들이 모여 있습니다.	2009-06-23
천문	천문에 관련된 커뮤니티들이 모여 있습니다.	2009-06-23
취미a	취미에 관련된 커뮤니티들이 모여 있습니다.a	2009-06-23
Page 1 of 3 View Count : 1 - 5 of 12		
추가 삭제 저장		

- 추가 버튼을 클릭하면 Grid에 Row가 추가되고 신규 생성할 카테고리 정보를 입력할 수 있다.
- 기존 카테고리 정보를 Grid에서 직접 수정할 수 있다.
- Grid에서 삭제할 카테고리 선택 후 '삭제'버튼을 누르면 해당 Row의 삭제가 가능하다.
- '저장'버튼을 누를 시 사용자가 추가하거나 수정한 내용이 저장된다.
- '삭제'버튼을 누를 시 선택되어 있는 row의 정보를 삭제한다.
- 검색 기능 수행 시 검색 결과가 Grid에 표시된다.
- Grid 하단의 pagination을 통해 페이징 처리가 가능하다. 페이지 숫자를 직접 입력할 수 있으며, 또한 이전/다음 페이지 버튼 클릭으로 이동하는 것도 가능하다.

## 9.Grid(jQgrid) + Form

### 9.1.Grid + Form 화면 소개

본 Grid + Form 화면은 아래와 같은 UI 및 기능을 제공한다

- UI : Grid(jqGrid), Grid Pagination, Button
- 기능 : 검색, 게시물 추가/삭제/조회/수정

### 9.2.Grid + Form 화면

커뮤니티에 등록된 게시물 목록 조회 및 관리하는 화면이다. Grid 하단에는 Pagination이 존재한다.

BOARD GRID EXAMPLE

제목	등록자	등록일
엔진 청소	js.park	2009-09-24
유아용 책상	hong80	2009-08-25
컴퓨터 조립 가장 싸게	hong80	2009-08-25
조립 20일만 하면	test	2009-08-21
수원 성대	kkw	2009-08-21

Page 1 of 6 View Count : 1 - 5 of 26

추가 삭제 저장

DETAIL CONTENT VIEW

게시글 ID	
제목	
내용	
등록자	
등록일	
커뮤니티	FLEX 쉽게 배우보기

- 목록에서 게시글을 선택하면 하단 입력 Form에서 해당 게시글에 대한 상세 내용을 조회하고 수정할 수 있다.
- '추가'버튼을 클릭 시 하단에 빈 입력 Form에서 신규 게시글을 작성할 수 있다.
- '삭제'버튼을 클릭 시 선택된 게시물이 삭제된다.
- '저장'버튼을 누를 시 사용자가 추가하거나 수정한 내용이 저장된다.
- 검색 기능 수행 시 검색 결과가 Grid에 표시된다.



- Grid 하단의 pagination을 통해 페이지징 처리가 가능하다. 페이지 숫자를 직접 입력할 수 있으며, 또한 이전/다음 페이지 버튼 클릭으로 이동하는 것도 가능하다.

# 10.Tree Grid(jqgrid)

## 10.1.Tree Grid 화면 소개

본 TreeGrid 화면은 아래와 같은 UI 및 기능을 제공한다.

- UI : TreeGrid(jqGrid), Grid(jqGrid), Dialog, Tab, Autocomplete, Button
- 기능 : 검색, 부서 목록 조회, 사용자 조회/추가/수정/삭제

## 10.2.Tree Grid 화면

부서 목록을 TreeGrid(jqGrid)로 조회하고, 부서에 속한 사용자를 관리하는 화면이다. TreeGrid 및 Grid의 사용 방법을 제시하고 있다. TreeGrid의 검색을 돕기 위한 검색어 자동 완성 기능(AutoComplete)이 구현되어 있으며 그 외로 사용자 정보를 등록하고 수정하기 위한 팝업창(Dialog) 및 Tab(Tab)기능 또한 구현되어 있다.

User Grid Example

부서명 :

검색

이름	설명
영업그룹	영업그룹입니다.
▼ 국내영업팀	국내영업팀입니다.
● 국내영업파트	국내영업파트입니다.
▼ 해외영업팀	해외영업팀입니다.
● 유럽총괄파트	유럽총괄파트입니다.
● 아시아총괄파트	아시아총괄파트입니다.
▶ 개발그룹	개발그룹입니다.
▶ RnD그룹	RnD그룹입니다.
▶ 총무그룹	총무그룹입니다.
▶ 인사그룹	인사그룹입니다.

이름	직급	전화	휴대전화
김경호	부장	010-0808-0808	010-0808-0808
박성욱	과장	82-031-123-1234	82-010-123-1234
이경진	사원	02-123-4567	010-123-4567
이동우	과장	010-6366-9999	010-6366-9999
이유리	사원	041-529-5294	010-529-5294

추가

삭제

- 좌측 TreeGrid에서 부서를 선택하면 우측 Grid에 해당 부서에 속한 사용자 목록이 나열된다.
- 우측 Grid 내의 사용자를 선택 후 더블클릭 시 사용자에 대한 상세정보를 조회할 수 있는 팝업창(Dialog)이 나타난다. 이 창을 통하여 사용자 정보를 조회 및 수정할 수 있다.

- 우하단의 '추가'버튼을 클릭 시 사용자 추가가 가능한 팝업창(Dialog)이 나타난다. 이 창을 통하여 사용자 정보를 삭제할 수 있다.
- 우하단의 '삭제'버튼을 클릭 시 사용자 삭제가 가능하다. 단, Grid에서 삭제할 사용자를 지정한 경우에만 가능하며 지정하지 않았을 시에는 사용자를 선택하라는 메시지를 나타내며 삭제되지 않는다.
- 팝업창의 '저장'버튼을 클릭 시 해당 사용자 정보 저장이 가능하다. 이 것을 통하여 사용자 정보를 추가 및 수정할 수 있다.
- 팝업창의 '취소'버튼을 클릭 시 해당 팝업창(Dialog)을 종료할 수 있다.
- 팝업창(Dialog)에는 tab이 적용되어 있으며, Tab기능을 사용하여 분류(기본정보 추가정보)별로 사용자 정보를 입력/조회할 수 있다.
- Tree Grid 검색 시 자동 완성 기능을 제공한다. 검색 기능 수행 시 Tree Grid에 해당되는 row가 선택된다.



### Tree Grid 데이터 처리

Tree Grid는 서버로부터 받아오는 데이터의 sorting기능을 제공하지 않는다. 이에 서버에서 정렬된 데이터를 받아오게끔 하는 것이 중요하다. 본 Sample UI에서는 각각의 데이터의 ID가 Tree순으로 정렬되어있는 DB Table을 이용하였다.

# 11.Tree Grid(jqgrid) + Form

## 11.1.Tree Grid + Form 화면 소개

본 TreeGrid + Form 화면은 아래와 같은 UI 및 기능을 제공한다.

- UI : TreeGrid(jqGrid), Autocomplete, Button
- 기능 : 검색, 카테고리/커뮤니티 목록 조회, 커뮤니티 조회/수정

## 11.2.Tree Grid + Form 화면

커뮤니티(카테고리 포함) 목록을 TreeGrid(jqGrid)로 조회하고, 해당 커뮤니티 정보를 조회하는 화면이다. TreeGrid의 사용 방법을 제시하고 있으며 TreeGrid의 검색을 돕기 위한 검색어 자동 완성 기능(AutoComplete)이 구현되어 있다.

COMMUNITY GRID EXAMPLE

커뮤니티 이름 :

커뮤니티명	커뮤니티
▼ 컴퓨터	커뮤니티 ID COMMUNITY-0012
● JAVA 개발정보 나누기	커뮤니티 이름 컴퓨터 만들기
● THE PRACTICAL SQL	상세설명 컴퓨터 조립해서 써요~
● HTML CSS 자바스크립트	
● FLEX 쉽게 배우보기	
● Spring Framework 사용자 모임	
● 컴퓨터 만들기	
▼ 자동차	
● SM3 같이 타고	
● 중고차 싸게 팔고 차게 사기	
● 자동차 보험에 대한 모든 것	
● 혼자서 자동차 고치기	
● 어느 주유소에서 기름 넣으세요?	
● 자동차 함께 타기	
▼ 스포츠	
● MLB 매니아	
● K리그 봐요	
● 연원조기축구	
▼ 천문	
● 동기모임	
▼ 취미a	
● 목공 나들이	

- 좌측 TreeGrid에서 커뮤니티를 선택하면 우측에 커뮤니티 정보가 나타난다.
- 좌측 TreeGrid에서 카테고리에 대한 정보는 우측화면에서 조회할 수 없다.
- '저장'버튼을 클릭하여 수정한 커뮤니티 정보의 저장이 가능하다.
- 카테고리에 대한 정보 수정은 불가능하다.

- Tree Grid 검색 시 자동 완성 기능을 제공한다.

---

# 12.기타

## 12.1.참고 사이트

- jQuery : <http://api.jquery.com/>
- jQuery UI : <http://jqueryui.com> [<http://jqueryui.com/>]
- jqGrid : <http://www.trirand.com> [<http://www.trirand.com/>]