rMate Grid for HTML5 사용 설명서

Version 6.0

정품을 구입하신 고객에게는 기술상담 및 지원을 제공합니다

서울시 영등포구 영신로 220, 611 호

(영등포동 8 가, KnK 디지털타워)

(Tel : 02-2655-9767, riamore@riamore.net)

㈜리아모어소프트

목 차

1. 개요

1.1. rMate Grid 의 주요 특징

rMate Grid 는 ActiveX 기술을 배제한 순수 자바스크립트를 바탕으로 개발된 어플리케이션으로 사용자에게 데이터를 시각적으로 알기 쉽게 제시할 수 있는 솔루션을 제공합니다.
JSP, .NET, PHP, ASP, ColdFusion 등과 같은 웹 스크립트 언어에 구애 받지 않고, 그리드를 생성할 수 있습니다.
http://www.riamore.net 의 데모 메뉴에서 강력하고 쉬운 그리드를 직접 체험하실 수 있습니다.

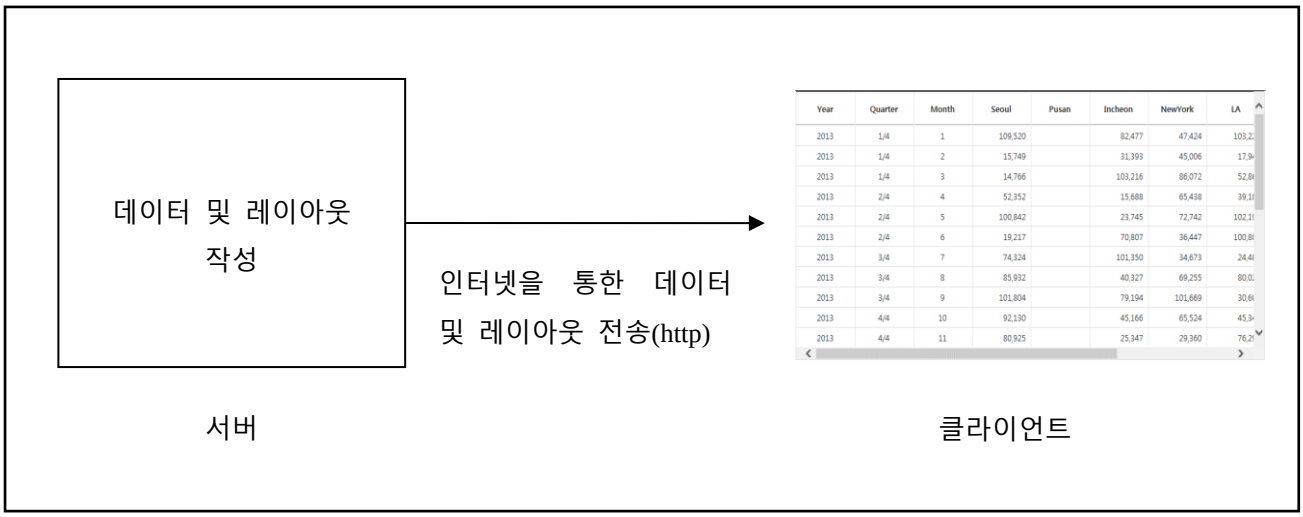
1.2. 시스템 요구사항

- 서버 : rMate Grid 는 순수 자바스크립트를 바탕으로 두기 때문에 톰캣, IIS, 웹로직, 웹스피어, 제우스, LCDS 등등 모든 WAS 서버에서 작동하며, 웹 스크립트 언어에 의존적이지 않습니다.
- 클라이언트 : 아래 표의 브라우저 버전을 만족하는 모든 시스템에서 가능합니다.

크롬	파이어폭스	사파리	엣지	아이폰	안드로이드
3.0+	31.0+	7.0+	80.0+	6.0+	4.0+

※ 6.0 버전부터 IE 에 지원은 중단됩니다. IE 를 사용하셔야 할 경우 5.0 이하 버젼을 사용하시기 바랍니다.

1.3. 그리드 시스템 구성 환경 설명



<그림 1 서버와 클라이언트 간의 구성도>

rMate Grid 를 생성하기 위해서는 그리드 레이아웃과 그리드에 표현하고자 하는 데이터가 필요합니다. 그리드 레이아웃은 그리드를 어떻게 표현할지를 나타내는 서식을 결정하게 됩니다. 예를 들면 그리드의 형태, 셀의 색상 및 병합 유무 등등 세밀한 부분을 제어하는 XML 파일입니다.
rMate 그리드 데이터 연동은 다음 5 가지 형태를 지원합니다.

- URL 을 호출하여 XML 데이터를 받아 처리하는 형태
- URL 을 호출하여 JSON 데이터를 받아 처리하는 형태
- XML 문자열을 자바스크립트를 통해 전달하는 형태
- JSON 문자열을 자바스크립트를 통해 전달하는 형태
- 자바스크립트에서 배열을 선언하고 값을 넣어 array 객체를 전달하는 형태.

이에 대한 자세한 설명은 제공된 CD 의 샘플을 실행시켜 다음 항목에서 확인할 수 있습니다.

제품 CD의 index.html 실행 → 기능별 분류 → 그리드 외형/데이터 설정

1.4. 제품의 구성요소

제공된 rMate Grid for HTML5 CD 의 내용은 아래와 같습니다.
디렉토리 구조는 rMateGridH5, LicenseKey, Docs, Samples 으로 구성되어 있으며 각각 디렉토리 역할은 아래와 같습니다.

가. rMateGridH5
제품의 자바스크립트(rMateGridH5.js 등) 및 필요한 이미지가 위치하고 있으며, 실제 제품을 실행하는데 필요한 최소한의 파일이 들어 있습니다.
하위의 JS 디렉토리에는 제품에서 필요한 자바스크립트가 들어있으며, Assets 디렉토리에는 그리드에서 사용하는 이미지와 그에 따른 CSS 가 들어 있습니다. 적용시 rMateGridH5 디렉토리를 전체로 서버에 올리고 해당 url 을 적용하시면 작업이 편리합니다.

나. LicenseKey
rMate Grid 라이선스가 있는 폴더입니다. (rMateGridH5License.js)

다. Docs
설명서는 PDF 파일 형태로 제공됩니다.

라. Samples
각종 예제를 볼 수 있도록 작성된 html 과 xml 파일입니다.

2. 기본적인 rMate Grid 생성

2.1. rMate Grid 라이선스 등록하기

정상적인 그리드를 생성하기 위해서는 rMate Grid 라이선스를 등록하셔야 합니다. 제공된 시디의 다음경로에 있는 파일이 그리드에 삽입할 라이선스입니다.

- /LicenseKey/rMateGridH5License.js

위 자바스크립트 파일을 <head> 태그 안에 삽입하시면 됩니다. **다른 작업을 하실 필요는 없습니다.**

```
<!DOCTYPE html>
<html>
<head>

                                                …
                                                …
                                                …
<!—rMate Grid 라이선스 등록 완료 모습 -->
<script src="rMateGridH5License.js" language="javascript"></script>

<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">
                                                …
                                                …
                                                …
```

<예제 1 rMate Grid 라이선스 등록 예>

2.2. 제공된 샘플로 그리드 생성하기

그리드를 생성하기 위해서는 제공된 CD 의 다음 경로를 따라 각각의 파일을 복사하여 작업 디렉토리에 복사하십시오.

- 그리드 JS 파일 : rMateGridH5/JS/rMateGridH5.js
- 라이선스 파일 : LicenseKey/rMateGridH5License.js
- 그리드 CSS 파일 : rMateGridH5/Assets/rMateH5.css

위 파일을 다음과 같이 html 파일에 설정합니다.

```
<!DOCTYPE html>
<html>
<head>
<title>rMateGridH5 (RiaMore Soft)</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="Content-Script-Type" content="text/javascript" />
<meta http-equiv="Content-Style-Type" content="text/css" />

<!-- rMateGridH5 CSS -->
<link rel="stylesheet" type="text/css" href="../rMateGridH5/Assets/rMateH5.css"/>
<!-- rMateGridH5 라이센스 -->
<script type="text/javascript"
src="../LicenseKey/rMateGridH5License.js"></script>
<!-- rMateGridH5 라이브러리 -->
<script type="text/javascript" src="../rMateGridH5/JS/rMateGridH5.js"></script>

<script type="text/javascript">
// ---------------------- 그리드 설정 시작 ------------------------------------

// rMateGridH5에서 그리드 생성 준비가 완료될 경우 호출할 함수를 지정합니다.
var jsVars = "rMateOnLoadCallFunction=gridReadyHandler";

// rMateDataGrid 를 생성합니다.
// 파라메터 (순서대로)
//  1. 그리드의 id ( 임의로 지정하십시오. )
//  2. 그리드가 위치할 div 의 id (즉, 그리드의 부모 div 의 id 입니다.)
//  3. 그리드 생성 시 필요한 환경 변수들의 묶음인 jsVars
//  4. 그리드의 가로 사이즈 (생략 가능, 생략 시 100%)
//  5. 그리드의 세로 사이즈 (생략 가능, 생략 시 100%)
rMateGridH5.create("grid1", "gridHolder", jsVars, "100%", "100%");

// 그리드의 속성인 rMateOnLoadCallFunction 으로 설정된 함수.
// rMate 그리드의 준비가 완료된 경우 이 함수가 호출됩니다.
// 이 함수를 통해 그리드에 레이아웃과 데이터를 삽입합니다.
// 파라메터 : id - rMateGridH5.create() 사용 시 사용자가 지정한 id 입니다.
function gridReadyHandler(id) {
	document.getElementById(id).setLayout(layoutStr);
	document.getElementById(id).setData(gridData);
}

//---------------------- 그리드 설정 끝 ----------------------@nd

var layoutStr =
'<rMateGrid>\
	<NumberFormatter id="numfmt" useThousandsSeparator="true"/>\
	<PercentFormatter id="percfmt"/>\
	<DataGrid id="dg1" sortableColumns="true" headerHeight="50" rowHeight="30"
draggableColumns="true" showHeaders="true" horizontalScrollPolicy="auto"
variableRowHeight="false" selectionMode="singleRow">\
		<columns>\
			<DataGridColumn id="dg1col1" dataField="Year"
textAlign="center"/>\
			<DataGridColumn id="dg1col2" dataField="Quarter"
textAlign="center"/>\
```

```
                    <DataGridColumn id="dg1col3" dataField="Month"
textAlign="center"/>\
                    <DataGridColumn id="dg1col4" dataField="Seoul"
textAlign="right" formatter="{numfmt}"/>\
                    <DataGridColumn id="dg1col5" dataField="Pusan"
textAlign="right" formatter="{numfmt}"/>\
                    <DataGridColumn id="dg1col6" dataField="Incheon"
textAlign="right" formatter="{numfmt}"/>\
                    <DataGridColumn id="dg1col7" dataField="NewYork"
textAlign="right" formatter="{numfmt}"/>\
                    <DataGridColumn id="dg1col8" dataField="LA" textAlign="right"
formatter="{numfmt}"/>\
                    <DataGridColumn id="dg1col9" dataField="Washington"
textAlign="right" formatter="{numfmt}"/>\
                    <DataGridColumn id="dg1col10" dataField="Revenue"
textAlign="right" formatter="{numfmt}"/>\
                    <DataGridColumn id="dg1col11" dataField="Percent" width="80"
textAlign="right" formatter="{percfmt}"/>\
            </columns>\
      </DataGrid>\
</rMateGrid>';


var gridData = [{"Year":2007, "Quarter":"1/4", "Month":1, "Seoul":109520,
"Pusan":40454, "Incheon":82477, "NewYork":47424, "LA":103225, "Washington":61161,
"Revenue":444260, "Percent":40},
{"Year":2007, "Quarter":"1/4", "Month":2, "Seoul":15749, "Pusan":29714,
"Incheon":31393, "NewYork":45006, "LA":17945, "Washington":90148,
"Revenue":229956, "Percent":20},
{"Year":2007, "Quarter":"1/4", "Month":3, "Seoul":14766, "Pusan":97314,
"Incheon":103216, "NewYork":86072, "LA":52863, "Washington":93789,
"Revenue":448020, "Percent":40},
{"Year":2007, "Quarter":"2/4", "Month":4, "Seoul":52352, "Pusan":56859,
"Incheon":15688, "NewYork":65438, "LA":39181, "Washington":109514,
"Revenue":339031, "Percent":31},
{"Year":2007, "Quarter":"2/4", "Month":5, "Seoul":100842, "Pusan":30391,
"Incheon":23745, "NewYork":72742, "LA":102195, "Washington":30407,
"Revenue":360322, "Percent":33},
{"Year":2007, "Quarter":"2/4", "Month":6, "Seoul":19217, "Pusan":75298,
"Incheon":70807, "NewYork":36447, "LA":100805, "Washington":84934,
"Revenue":387508, "Percent":36},
{"Year":2007, "Quarter":"3/4", "Month":7, "Seoul":74324, "Pusan":64947,
"Incheon":101350, "NewYork":34673, "LA":24486, "Washington":57781,
"Revenue":357561, "Percent":28},
{"Year":2007, "Quarter":"3/4", "Month":8, "Seoul":85932, "Pusan":95733,
"Incheon":40327, "NewYork":69255, "LA":80024, "Washington":102739,
"Revenue":474011, "Percent":37},
{"Year":2007, "Quarter":"3/4", "Month":9, "Seoul":101804, "Pusan":65098,
"Incheon":79194, "NewYork":101669, "LA":30608, "Washington":73020,
"Revenue":451393, "Percent":35},
{"Year":2007, "Quarter":"4/4", "Month":10, "Seoul":92130, "Pusan":91881,
"Incheon":45166, "NewYork":65524, "LA":45348, "Washington":72708,
"Revenue":412757, "Percent":36},
{"Year":2007, "Quarter":"4/4", "Month":11, "Seoul":80925, "Pusan":70537,
"Incheon":25347, "NewYork":29360, "LA":76296, "Washington":42766,
"Revenue":325230, "Percent":29},
{"Year":2007, "Quarter":"4/4", "Month":12, "Seoul":99008, "Pusan":30598,
"Incheon":99124, "NewYork":22776, "LA":107805, "Washington":38384,
"Revenue":397696, "Percent":35},
{"Year":2008, "Quarter":"1/4", "Month":1, "Seoul":68503, "Pusan":10155,
"Incheon":47908, "NewYork":60857, "LA":104179, "Washington":109097,
```

```
"Revenue":400699, "Percent":31},
{"Year":2008, "Quarter":"1/4", "Month":2, "Seoul":80573, "Pusan":75743,
"Incheon":107750, "NewYork":76243, "LA":79265, "Washington":85345,
"Revenue":504918, "Percent":40},
{"Year":2008, "Quarter":"1/4", "Month":3, "Seoul":23435, "Pusan":30538,
"Incheon":86528, "NewYork":36735, "LA":96031, "Washington":96928,
"Revenue":370196, "Percent":29},
{"Year":2008, "Quarter":"2/4", "Month":4, "Seoul":35657, "Pusan":109415,
"Incheon":45569, "NewYork":87683, "LA":92773, "Washington":53422,
"Revenue":424520, "Percent":45},
{"Year":2008, "Quarter":"2/4", "Month":5, "Seoul":50140, "Pusan":30142,
"Incheon":83992, "NewYork":87292, "LA":72324, "Washington":32520,
"Revenue":356410, "Percent":37},
{"Year":2008, "Quarter":"2/4", "Month":6, "Seoul":39458, "Pusan":10848,
"Incheon":10553, "NewYork":48474, "LA":25642, "Washington":36591,
"Revenue":171565, "Percent":18},
{"Year":2008, "Quarter":"3/4", "Month":7, "Seoul":33761, "Pusan":49046,
"Incheon":31351, "NewYork":46829, "LA":97148, "Washington":42630,
"Revenue":300765, "Percent":31},
{"Year":2008, "Quarter":"3/4", "Month":8, "Seoul":89645, "Pusan":72565,
"Incheon":23678, "NewYork":78847, "LA":62559, "Washington":87722,
"Revenue":415017, "Percent":42},
{"Year":2008, "Quarter":"3/4", "Month":9, "Seoul":14844, "Pusan":30709,
"Incheon":83037, "NewYork":23130, "LA":65006, "Washington":48367,
"Revenue":265093, "Percent":27},
{"Year":2008, "Quarter":"4/4", "Month":10, "Seoul":30598, "Pusan":55523,
"Incheon":90576, "NewYork":79997, "LA":71346, "Washington":63569,
"Revenue":391608, "Percent":34},
{"Year":2008, "Quarter":"4/4", "Month":11, "Seoul":64461, "Pusan":61341,
"Incheon":74479, "NewYork":10715, "LA":40404, "Washington":93611,
"Revenue":345011, "Percent":30},
{"Year":2008, "Quarter":"4/4", "Month":12, "Seoul":99229, "Pusan":95468,
"Incheon":108828, "NewYork":27176, "LA":28673, "Washington":54816,
"Revenue":414191, "Percent":36}];
</script>

</script>
</head>
<body>
	<div class="content">
		<!-- 그리드가 삽입될 DIV -->
		<div id="gridHolder">
		</div>
	</div>
</body>
</html>
```

<예제 2 HTML 페이지에 그리드 삽입하기>

위 처럼 정의하여 저장 후 실행시키면 다음과 같은 화면을 볼 수 있습니다.

Year	Quarter	Month	Seoul	Pusan	Incheon	NewYork	LA
2013	1/4	1	109,520		82,477	47,424	103,2
2013	1/4	2	15,749		31,393	45,006	17,9
2013	1/4	3	14,766		103,216	86,072	52,8
2013	2/4	4	52,352		15,688	65,438	39,1
2013	2/4	5	100,842		23,745	72,742	102,1
2013	2/4	6	19,217		70,807	36,447	100,8
2013	3/4	7	74,324		101,350	34,673	24,4
2013	3/4	8	85,932		40,327	69,255	80,0
2013	3/4	9	101,804		79,194	101,669	30,6
2013	4/4	10	92,130		45,166	65,524	45,3
2013	4/4	11	80,925		25,347	29,360	76,2

<그림 2 HTML 페이지에 그리드 삽입 성공 화면>

2.3. 그리드 생성시 유의 사항

사용자가 웹 페이지를 작성할 때 지켜야 할 사항은 다음과 같습니다.

1. HTML 의 doctype 을 지정하십시오. (다른 doctype 이어도 무방합니다)

```
<!DOCTYPE html>
```

2. rMateGridH5License.js 와 rMateGridH5.js 를 포함시켜 주십시오.

```
<script language="javascript" type="text/javascript"
src="../LicenseKey/rMateGridH5License.js"></script>
<script language="javascript" type="text/javascript"
src="../rMateGridH5/JS/rMateGridH5.js"></script>
```

※ 만일 html 문서 인코딩을 utf-8 이 아닌 다른 문자셋을 사용할 경우에는 다음과 같이 자바스크립트를 포함시키는 문장에 charset 을 utf-8 로 설정해 주시기 바랍니다. (자바스크립트는 utf-8 로 인코딩 되어 있음)

```
<script type="text/javascript" src="../rMateGridH5/JS/rMateGridH5.js" charset="utf-8"></script>
```

3. 그리드에서 사용할 이미지의 CSS 를 포함시켜 주십시오.

```
<link rel="stylesheet" type="text/css" href="../rMateGridH5/Assets/rMateH5.css"/>
```

※ 로딩되는 css 는 rMate Grid 에서 접근하는 경우가 있는데 크롬, 파이어폭스에서는 다른 서버에서 읽어들인 css 에 대해서는 조작시 보안 문제가 발생할 수 있으니 되도록 같은 서버에서 css 를

읽어들이는 것을 권장합니다.

4. 그리드가 생성될 Div 태그를 지정하여 영역을 확보해 주십시오.

```
<div id="gridHolder" style="width:600px; height:400px;"></div>
```

※ height 를 100%로 줄 경우 브라우저에서 계산이 안되어 크기가 0 으로 잡힐 수 있으므로 유의하시기 바랍니다.

5. 그리드의 레이아웃과 데이터를 작성하여 주십시오.

6. rMateGridH5.create() 함수 호출로 그리드를 생성합니다.

```
// rMateGrid 를 생성합니다.
// 파라메터 (순서대로)
//  1. 그리드의 id ( 중첩이 안되도록 임의로 지정하십시오. )
//  2. 그리드가 위치할 div 의 id (즉, 그리드의 부모 div 의 id 입니다.)
//  3. 그리드 생성 시 필요한 환경 변수들의 묶음인 jsVars
//  4. 그리드의 가로 사이즈 (생략 가능, 생략 시 100%)
//  5. 그리드의 세로 사이즈 (생략 가능, 생략 시 100%)
rMateGridH5.create("grid1", "gridHolder", jsVars, "100%", "100%");
```

3. 그리드 데이터 형식

rMate Grid 의 수치 데이터는 **XML 형식과 JSON 형식, CSV 또는 TSV 형식, 자바스크립트 배열 형식**을 지원합니다. 사용자는 몇 가지 규칙에 따라 그리드 데이터를 작성할 필요가 있습니다. 규칙에 의거하여 데이터 XML 형식과 JSON 형식, 배열형식으로 각각 변환하여 어떻게 그리드에 삽입하는지를 이번 장에서 설명합니다.

3.1. 데이터 형식 지정

rMate Grid 에서 데이터의 기본형식은 JSON 입니다. 따라서 JSON 이외의 형식을 사용할 경우에는 사용할 형식을 지정해 줘야 합니다.
사용할 데이터의 형식을 지정하는 방식은 jsVars 를 이용하는 방법(6.1 jsVars 설정 참조)과 setDataType 함수(API Reference 문서의 GridApp 클래스 참조)를 이용하는 방법이 있습니다.
자바스크립트 배열을 이용하여 데이터를 넣을 경우에는 별도로 dataType 을 지정해 주지 않아도 배열로 인식하여 작동합니다.

3.2. XML 형식의 데이터 작성 및 그리드에 적용하기

XML 형식으로 데이터를 작성 할 때 반드시 준수해야 할 점은 맨 처음의 XML root 가 존재해야 한다는 점 입니다. 따라서 rMate Grid root 밑의 children 을 가지고 데이터로 인식하여 출력하게 됩니다. 데이터 XML 파일이 여러 root 를 가지게 되면 그리드는 데이터를 표현하지 않습니다.

매출(Revenue), 비용(Cost), 이윤(Profit)을 바탕으로 월간 보고서를 그리드로 표현하기 위해 데이터를 작성해 보도록 하겠습니다.

월(Month)	매출(Revenue)	비용(Cost)	이윤(Profit)
Jan	10,000	5,000	5,000
Feb	15,000	7,000	8,000
Mar	12,000	6,000	6,000
Apr	30,200	4,000	26,200
May	28,000	10,000	18,000
Jun	12,000	5,000	7,000
Jul	22,000	10000	12,000
Aug	13,000	6,000	7,000
Sep	22,000	10,000	12,000
Oct	29,000	8,000	11,000

Nov	18,000	7,500	10,500
Dec	30,000	12,000	28,000

<XML 형식의 데이터>

```
<Sales>
        <item>
            <Month>Jan</Month>
            <Revenue>10000</ Revenue >
            <Cost>5000</Cost>
            <Profit>5000</Profit>
        </item>
        <item>
            <Month>Feb</Month>
            <Revenue>15000</Revenue>
            <Cost>7000</Cost>
            <Profit>8000</Profit>
        </item>
                        :
                        :
        <item>
            <Month>Dec</Month>
            <Revenue>30000</Revenue>
            <Cost>12000</Cost>
            <Profit>18000</Profit>
        </item>
</Sales>
```

<예제 3 XML 형식으로 데이터 변환>

위와 같이 데이터를 XML 유형으로 표현할 수 있습니다.
예를 들어 위에 작성한 XML 데이터를 "singleData.xml" 로 저장하였다면 그리드에 singleData.xml 의 경로를 넘겨줘야 합니다. 이에 대한 예제는 <예제 2 HTML 페이지에 그리드 삽입하기> 를 참고하세요.

* XML 로 데이터를 작성할 경우 소팅시 숫자에 의한 소팅이 아닌 문자열에 의한 소팅이 되므로 숫자 컬럼에 대한 소팅기능을 사용해야 할 경우에는 컬럼의 sortCompareFunction 에 numericSort 를 설정해 줘야 합니다.(샘플 -> 그리드 기본기능 -> 컬럼정렬(XML 데이터 사용시) 참조) 또한 자바스크립트에서 이벤트나 함수를 통해 전달되는 데이터의 레코드들은 XML Element 형식으로 전달되니 유의하시기 바랍니다. (샘플 -> 그리드 외형/데이터 설정 -> XML 데이터 제어 참조)

3.3. JSON 형식의 데이터 작성 및 그리드에 적용하기

JSON 형식으로 데이터를 작성 할 때 파일의 형식은 http://www.json.org/ 에서 정의한 형식을 반드시 준수해야 하며 데이터 JSON 파일이 형식에 안 맞을 경우에는 그리드는 데이터를 표현하지 않습니다.
위의 3.1 에서 정의한 매출(Revenue), 비용(Cost), 이윤(Profit)을 바탕으로 월간 보고서를 그리드로 표현하기 위해 데이터를 작성해 보도록 하겠습니다.

<JSON 형식의 데이터>

```
[
    {
        "Month": "Jan",
        "Revenue": 10000,
        "Cost": 5000,
        "Profit": 5000
    },
    {
        "Month": "Feb",
        "Revenue": 15000,
        "Cost": 7000,
        "Profit": 8000
    },
                        .
                        .
                        .
    {
        "Month": "Dec",
        "Revenue": 30000,
        "Cost": 12000,
        "Profit": 18000
    }
]
```

<예제 4 JSON 형식으로 데이터 변환>

위와 같이 데이터를 JSON 유형으로 표현할 수 있습니다.

3.4. CSV, TSV 형식의 데이터 작성 및 그리드에 적용하기

CSV(comma-separated values) 또는 TSV(tab-separated values) 형식으로 데이터를 작성 할 때 파일의 형식은 http://en.wikipedia.org/wiki/Comma-separated_values 에서 설명한 형식을 기준으로 하며, 데이터 파일이 형식에 안 맞을 경우에는 그리드는 데이터를 표현하지 않습니다.

JSON 이나 XML 과 달리 각 데이터 필드의 필드명이 없으므로 기본으로 생성되는 필드명은 순서에 따라 F0,F1,F2,F3... 와 같이 앞에 "F"를 넣고 뒤에 순서에 따른 숫자가 따라 붙는 형식으로 생성되게 됩니다. 만일 그리드의 컬럼을 기준으로 필드명을 지정하고 싶으시면 gridRoot 의 useDataGridFields 속성을 true 로 주고 그리드의 레이아웃이 완성되어 dataGrid 가 생성된 후에 데이터를 설정하시면 컬럼의 dataField 명에 따라 데이터가 생성됩니다. 또한 데이터의 크기가 10Mbytes 이상일 경우에 자바스크립트 timeout(수행시간 초과)을 피하기 위해 비동기 모드로 파싱이 일어나게 되어 파싱 속도가 느려질 수 있습니다.
CSV,TSV 형식은 엑셀에서 로딩하여 테스트해 보실 수 있습니다. (TSV 는 txt 파일로 저장으로 테스트 가능)
CSV 나 TSV 형식을 사용할 경우에는 XML 이나 JSON 보다 형식이 간단하므로 대용량 데이터를 처리해야 할 경우 로딩 속도를 개선 시킬 수 있습니다. 또한 이를 위해 fast 모드를 지원합니다.
CSV 일 경우 fast 모드는 dataType 속성에 "fastCsv"로 지정하여 사용하실 수 있는데, 필드의 문자열에 따옴표(")가 안 들어가는 경우에 사용하면 파싱과정이 간단하게 되어 더욱 빠른 속도로 처리가 가능합니다. 하지만 이를 위해서는 데이터 안에 ',' 나 리턴키가 없어야 합니다.
마찬가지로 TSV 일 경우에는 "fastTsv"를 지정하여 사용하실 수 있으며 필드의 문자열에 따옴표나 탭문자, 리턴키가 안 들어가는 경우에 사용하시면 됩니다.

3.5. 자바스크립트 배열 형식의 데이터 작성 및 그리드에 삽입하기

rMate Grid 는 사용자의 편의를 위하여 배열 형식의 데이터를 그리드에 삽입할 수 있도록 하였습니다. 그 방법에 대한 설명입니다.
매월 이윤에 대해 자바스크립트에서 배열 형식으로 데이터를 작성해 보도록 하겠습니다.

월(Month)	이윤(Profit)
Jan	10,000
Feb	15,000
Mar	12,000
Apr	30,200
May	28,000
Jun	12,000
Jul	22,000
Aug	13,000
Sep	22,000
Oct	29,000
Nov	18,000
Dec	30,000

```
var gridData = [{"Month":"Jan", "Profit":10000},
                {"Month":"Feb", "Profit":15000},
                {"Month":"Mar", "Profit":12000},
                {"Month":"Apr", "Profit":30200},
                {"Month":"May", "Profit":28000},
                {"Month":"Jun", "Profit":12000},
                {"Month":"Jul", "Profit":22000},
                {"Month":"Aug", "Profit":13000},
                {"Month":"Sep", "Profit":22000},
                {"Month":"Oct", "Profit":29000},
                {"Month":"Nov", "Profit":18000},
                {"Month":"Dec", "Profit":30000} ];
```

<예제 5 배열형식으로 데이터 변환>

위와 같이 배열 형식으로 변환할 수 있습니다.

3.2 장에서 작성된 매출(Revenue), 비용(Cost), 이윤(Profit)을 바탕으로 작성된 다중 데이터를 배열형식으로 변환하면 다음과 같습니다.

```
var gridData = [{"Month":"Jan", "Revenue":10000, "Cost":5000, "Profit":5000},
                {"Month":"Feb", "Revenue":15000, "Cost":7000, "Profit":8000},
                {"Month":"Mar", "Revenue":12000 , "Cost":6000, "Profit":6000},
                {"Month":"Apr", "Revenue":30200, "Cost":4000, "Profit":26200},
                {"Month":"May", "Revenue":28000, "Cost":10000, "Profit":18000},
                {"Month":"Jun", "Revenue":12000, "Cost":5000, "Profit":7000},
                {"Month":"Jul", "Revenue":22000, "Cost":10000, "Profit":12000},
                {"Month":"Aug", "Revenue":13000, "Cost":6000, "Profit":7000},
                {"Month":"Sep", "Revenue":22000, "Cost":10000, "Profit":12000},
                {"Month":"Oct", "Revenue":29000, "Cost":8000, "Profit":21000},
```

```
{"Month":"Nov", "Revenue":18000, "Cost":7500, "Profit":10500},
{"Month":"Dec", "Revenue":30000, "Cost":12000, "Profit":18000} ];
```

<예제 6 배열형식으로 다중 데이터 변환>

3.6. 날짜 형식의 자료

데이터의 형식에 관계없이 날짜 데이터를 넣는 경우에는 문자열로 넣어지며, 기본으로 자바스크립트의 Date.parse 함수를 사용하여 파싱하기 때문에 날짜 객체로 변환시 오류가 발생할 수 있습니다.
컴포넌트중 날짜와 관련된 DateFormatter 나 DateEditor 등은 전달된 데이터를 날짜형식으로 변환합니다. 브라우저 별로 날짜의 형식이 다르게 적용되기 때문에(참조: http://dygraphs.com/date-formats.html) 기본으로는 아래 형식중의 하나여야 정상적으로 Date 객체로 변환되며, 형식이 잘못 입력된 경우에는 Date 파싱을 위한 포맷 문자열을 별도로 넣어주어야 오류가 발생하지 않습니다.

YYYY/MM/DD
YYYY/M/DD
YYYY/MM/DD HH:NN
YYYY/MM/DD HH:NN:SS
MM/DD/YYYY
M/D/YYYY
M/D/YYYY HH:NN

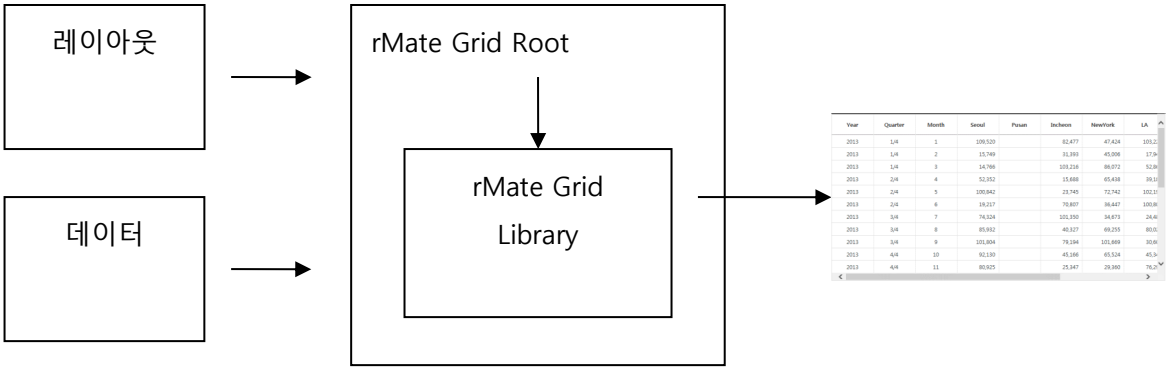
Date 파싱을 위한 포맷 문자열은 DateFormatter 의 경우 parseFormatString, DateEditor 의 경우 dataFormatString 속성이 있습니다.

- 버전 2.0 부터 YYYYMMDD 형식과 YYYY-MM-DD, YYYY.MM.DD 형식에 대해 기본 지원합니다 다만 YYYY-MM-DD 형식을 사용할 경우 time zone 에 의거한 시간 값이 기본으로 들어가게 되니 주의 하시기 바랍니다. (사파리 제외)

4. 그리드 레이아웃

4.1. rMate Grid 에서 레이아웃의 역할

rMate Grid 는 미리 작성된 레이아웃의 XML 을 파싱하여 동적으로 생성됩니다. 사용자가 작성한 레이아웃을 rMateGrid 에 입력하면 그리드는 미리 내포하고 있는 클래스를 기반으로 레이아웃에 작성된 그리드를 생성하고 데이터를 출력하게 됩니다.
다음은 이런 과정을 다이어그램으로 표현한 것입니다.



<그림 3 rMate Grid 동작 다이어그램>

4.2. 레이아웃 작성 방법에 대하여

레이아웃은 XML 형식으로 작성하여야 하므로 엘리먼트나 속성을 넣는 방식도 그에 적합하여야 하며 아래 형식에 따라 작성되어야 합니다.
rMate Grid 의 모든 레이아웃은 <rMateGrid>태그로 시작하여 </rMateGrid>태그로 끝납니다. 레이아웃은 크게 그리드 UI 요소와 그리드, Style 세 부분으로 구분됩니다. 그리드 UI 요소는 선택적으로 필요시 정의되는 요소이며, Style 또한 그리드의 스타일 제어를 위해 필요시 정의 하면 됩니다.

rMate Grid 는 ID 에 의한 객체 접근을 지원합니다. 예를 들어 DataGridColumn 객체의 속성으로 formatter 가 존재합니다. 이 formatter 속성의 속성값으로 그리드 UI 요소에서 정의한 NumberFormatter 인스턴스를 할당합니다. 이를 표현한 일반적인 방법은 아래와 같습니다.

```
<rMateGrid>
    <DataGrid>
            .
            .
            .
        <DataGridColumn dataField="Profit" textAlign="right">
            <formatter>
                <NumberFormatter useThousandsSeparator="true"/>
```

```
            </formatter>
        </DataGridColumn>
    </DataGrid>
</rMateGrid>
```

<예제 7 일반적인 속성 값 할당 법>

그러나 사용자의 편의를 위해 <예제 8 ID 에 의한 속성 값 할당 법>과 같이 NumberFormatter 인스턴스를 미리 생성한 후 그 ID 를 통해 NumberFormatter 인스턴스를 활용할 수 있습니다. 이는 여러 객체들이 하나의 NumberFormatter 인스턴스에 접근하고자 할 때 불필요하게 NumberFormatter 인스턴스를 생성 하지 않고 미리 만들어진 하나의 NumberFormatter 에 접근하기 때문에 효율적입니다. 또한 어느 특정 객체를 반드시 참조해야 하는 경우에도 효율적입니다.

```
<rMateGrid>
    <NumberFormatter id="numfmt" useThousandsSeparator="true"/>
    <DataGrid>
                .
                .
                .
                                    //객체 ID인 경우엔 시작과 끝을 중괄호({,})로 묶음
      <DataGridColumn dataField="Profit" textAlign="right" formatter="{numfmt}"/>
   </DataGrid>
</rMateGrid>
```

<예제 8 ID 에 의한 속성 값 할당 법>

4.3. 레이아웃의 UI 요소 설정하기

그리드 UI 요소는 레이아웃의 필수사항이 아닌 선택사항인 컴포넌트를 선언하는 것을 뜻합니다. 그리드에 사용될 ContextMenu(오른쪽 마우스 클릭시 나타나는 메뉴), 숫자나 날짜를 표현하는데 쓰이는 포맷터, 셀이나 행의 속성을 정의하는데 쓰이는 속성정의 컴포넌트 (SpanCellAttribute, SpanRowAttribute)등을 정의하면 됩니다.

```
<rMateGrid>
        <ContextMenu id="cMenu">
                <ContextMenuItem id="cMenuInsert" caption="Insert Row"/>
                <ContextMenuItem id="cMenuDelete" caption="Delete Row"/>
        </ContextMenu>
        <NumberFormatter id="numfmt" useThousandsSeparator="true"/>
        <DataGrid id="dg1" contextMenu="{cMenu}" horizontalScrollPolicy="auto"
horizontalGridLines="true" horizontalGridLineColor="#CCCCCC"
verticalAlign="middle" selectionMode="multipleRows">
                <columns>
                        <DataGridColumn dataField="Selected"/>
                        <DataGridColumn dataField="From"/>
```

```
                    <DataGridColumn dataField="Subject" />
                    <DataGridColumn dataField="Date"/>
                    <DataGridColumn dataField="CC"/>
                    <DataGridColumn dataField="Profit" textAlign="right"
formatter="{numfmt}"/>
                </columns>
        </DataGrid>
</rMateGrid>
```

<예제 9 그리드 UI 요소 사용 예>

4.4. 레이아웃의 특수 문자 처리

XML 의 특성상 태그의 attribute 안에 특정문자는 입력시 XML 파싱 오류를 발생하게 되므로 변환하여 입력해야 하며 그 내용은 다음과 같습니다.

- < : <
- > : >
- & : &
- " : "
- ' : '

또한 attribute 안에 대괄호([,])나, 중괄호({,})문자가 오면 JSON 형식으로 인식하여 파싱을 하게 되므로 headerText 나 text 와 같은 attribute 안에 해당 문자를 표시하고 싶을 때는 다음과 같이 변환하여 입력해야 제대로 표시되게 됩니다.

- { : {
- } : }
- [: [
-] :]

4.5. 레이아웃의 Style 노드(CSS 적용) 설정하기

<Style> 노드는 전반적인 스타일을 미리 정의하는 스타일시트이며, html 에서 사용하는 CSS 와 거의 동일한 정의를 사용합니다. <Style/>을 선언하는 방법에는 몇 가지 규칙이 있습니다.

첫째, <Style/> 노드는 반드시 <rMateGrid> 노드의 자식 위치에 정의하십시오.
둘째, 스타일 셀렉터를 정의 할 때 시작은 도트(.)을 찍고 소문자로 시작하는 영문자로 표기하십시오.
셋째, 스타일 셀렉터를 정한 후 그에 따른 구체적인 스타일을 정의 할 때는 반드시 중괄호({,})로 시작과 끝을 표시하십시오.
넷째, 속성명과 속성값(value)의 구분은 콜론(:)이며 끝은 세미콜론으로 나타냅니다.

다섯째, 속성명은 css 와 달리 CamelCase 형식으로 표기합니다 (background-color 가 아닌 backgroundColor 형식을 사용)

다음은 올바른 예와 잘못된 예를 나타낸 것입니다.

올바른 예	잘못 작성 된 예

<rMateGrid>
<DataGird>
 	
...
 	
<Style>
.rMateGridStyle
{
backgroundColor:#FFFFFE;
borderColor:#77EE9E;
borderWidth:3;
borderStyle:solid;
}
</Style>
</rMateGrid>
	<rMateGrid>
	<DataGrid>
	
	...
	
	<Style>
	#GridStyle
	[
	Background-color:#FFFFFE;
	BorderColor:"#77EE9E";
	borderWidth:3
	borderStyle:'solid';
	]
	</Style>
	</rMateGrid>

<표 1 Style 작성 예>

위와 같이 rMateGridStyle 스타일을 정의 하였습니다. 그렇다면 이 스타일을 적용하는 방법에 대해 알아보도록 하겠습니다.

```
<rMateGrid styleName="rMateGridStyle">
       <DataGrid>

        ......

       <Style>
       .rMateGridStyle
       {
       backgroundColor:#FFFFFE;
       borderColor:#77EE9E;
       borderRadius:12;
       borderWidth:3;
       borderStyle:solid;

       }
       </Style>

</rMateGrid>
```

<예제 10 Style 적용 예>

<예제 10 Style 적용 예>를 보시면 rMateGridStyle 은 <rMateGrid/> 노드의 스타일을 정의 한 것이고 적용법은 styleName="정의한 스타일이름" 즉, styleName="rMateGridStyle" 과 같습니다.
<Style/>에서 정의 가능한 각각의 속성은 API Reference 문서에서 각컴포넌트의 스타일 부분을 참조하시

면 됩니다.

스타일을 활용한 예제와 출력 모습은 아래와 같습니다.

```
<rMateGrid>
	<DataGrid id="dg1" styleName="gridStyle">
		<columns>
			<DataGridColumn id="dg1col1" dataField="Year"/>
			<DataGridColumn id="dg1col2" dataField="Quarter"/>
			<DataGridColumn id="dg1col3" dataField="Month"/>
			<DataGridColumn id="dg1col4" dataField="Seoul"/>
			<DataGridColumn id="dg1col5" dataField="Pusan"/>
			<DataGridColumn id="dg1col6" dataField="Incheon"/>
			<DataGridColumn id="dg1col7" dataField="NewYork"/>
			<DataGridColumn id="dg1col8" dataField="LA"/>
			<DataGridColumn id="dg1col9" dataField="Washington"/>
			<DataGridColumn id="dg1col10" dataField="Revenue"/>
			<DataGridColumn id="dg1col11" dataField="Percent"/>
		</columns>
	</DataGrid>
	<Style>
		.gridStyle {
		  color:#378F8B;
		  alternatingItemColors:#C5E3E3,#FFFFFF;
		  headerColors:#88C3C3,#5C8484;
		  headerStyleName:gridHeaderStyle;
		  horizontalGridLines:true;
		  horizontalGridLineColor:#5C8484;
		  selectionColor:#ADC1C1;
		  rollOverColor:#CC9999;
		  fontWeight:bold;
		  textDecoration:underline;
		  fontSize:12;
		  verticalAlign:middle;
		  verticalGridLineColor:#5C8484;
		}
	</Style>
</rMateGrid>
```

<예제 11 Style 적용한 예>

Year	Quar...	Month	Seoul	Pusan	Inch...	New...	LA	Was...	Reve...	Perc...
2007	1/4	1	109520	40454	82477	47424	103225	61161	444260	40
2007	1/4	2	15749	29714	31393	45006	17945	90148	229956	20
2007	1/4	3	14766	97314	103216	86072	52863	93789	448020	40
2007	2/4	4	52352	56859	15688	65438	39181	109514	339031	31
2007	2/4	5	100842	30391	23745	72742	102195	30407	360322	33
2007	2/4	6	19217	75298	70807	36447	100805	84934	387508	36
2007	3/4	7	74324	64947	101350	34673	24486	57781	357561	28
2007	3/4	8	85932	95733	40327	69255	80024	102739	474011	37
2007	3/4	9	101804	65098	79194	101669	30608	73020	451393	35
2007	4/4	10	92130	91881	45166	65524	45348	72708	412757	36
2007	4/4	11	80925	70537	25347	29360	76296	42766	325230	29
2007	4/4	12	99008	30598	99124	22776	107805	38384	397696	35
2008	1/4	1	68503	10155	47908	60857	104179	109097	400699	31
2008	1/4	2	80573	75743	107750	76243	79265	85345	504918	40
2008	1/4	3	23435	30538	86528	36735	96031	96928	370196	29
2008	2/4	4	35657	109415	45569	87683	92773	53422	424520	45
2008	2/4	5	50140	30142	83992	87292	72324	32520	356410	37
2008	2/4	6	39458	10848	10553	48474	25642	36591	171565	18

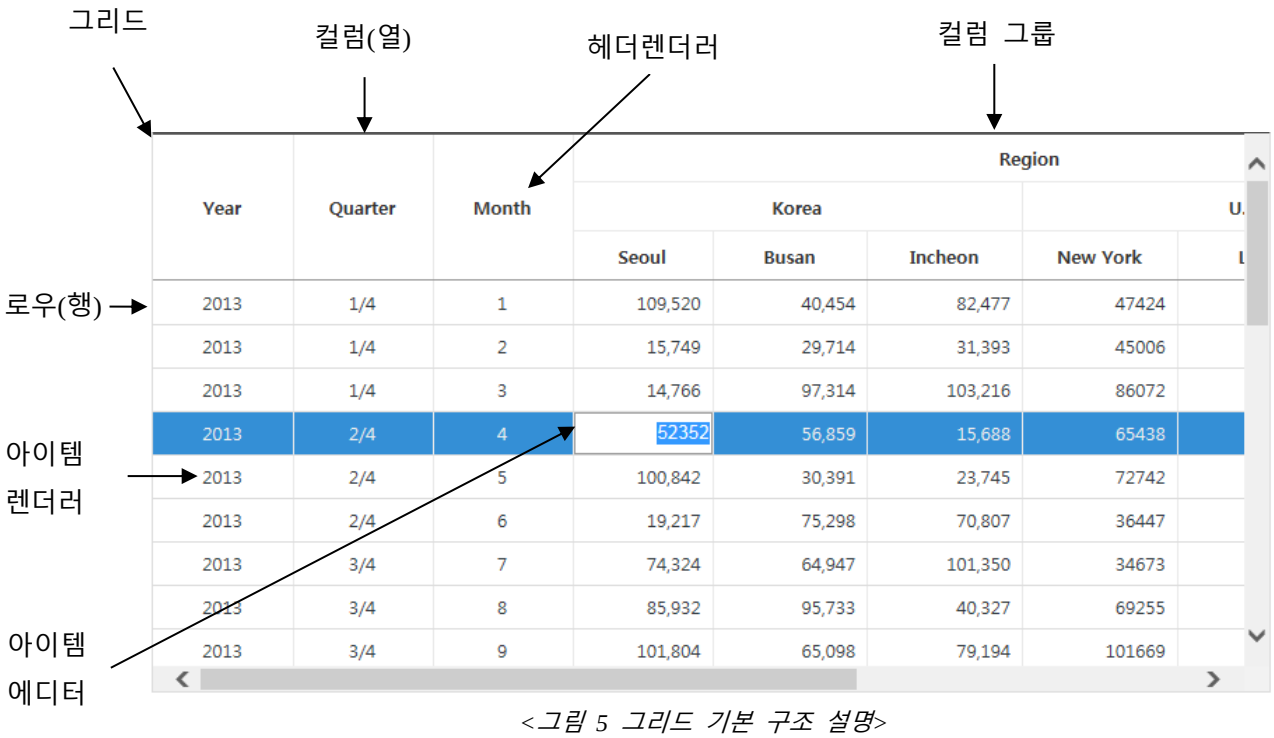
<그림 4 Style 적용 결과>

<예제 11 Style 적용 예>에서 alternatingItemColors 같이 색을 배열로 넣을 경우에는 <Style/> 안에서는 "["와 "]" 필요없이 ","로만 연결하여 표시하면 됩니다. 하지만

<DataGrid alternatingItemColors="[#FFCC33, #FFFF33]"> 와 같이 레이아웃 태그에서 처리하실 경우에는 "["와 "]"를 붙여서 배열처리를 해줘야 합니다.

5. 그리드의 기본 구조 및 용어설명

5.1. 그리드 기본구조 설명



Year	Quarter	Month	Region				
			Korea				U.
			Seoul	Busan	Incheon	New York	L
2013	1/4	1	109,520	40,454	82,477	47424	
2013	1/4	2	15,749	29,714	31,393	45006	
2013	1/4	3	14,766	97,314	103,216	86072	
2013	2/4	4	52352	56,859	15,688	65438	
2013	2/4	5	100,842	30,391	23,745	72742	
2013	2/4	6	19,217	75,298	70,807	36447	
2013	3/4	7	74,324	64,947	101,350	34673	
2013	3/4	8	85,932	95,733	40,327	69255	
2013	3/4	9	101,804	65,098	79,194	101669	

<그림 5 그리드 기본 구조 설명>

그리드의 기본 구조는 안에 컬럼이 존재하며 컬럼안에는 헤더 렌더러로 구성된 헤더부분과 아이템 렌더러로 구성된 셀이 있고 셀이 합쳐져 행을 이루는 바디 부분입니다.
필요에 따라 컬럼을 여럿 합쳐 그룹 컬럼을 만들 수 있습니다.
레이아웃 엘리먼트에 의해 설명한다면 <DataGrid> 노드에 의해 그리드가 만들어 지고 <DataGridColumn> 노드로 컬럼에 대한 정의를 설정하면, 그리드에서는 컬럼정보에 의해 헤더렌더러를 만들어 헤더부분을 채우고 데이터를 받아들여 각 컬럼정보에 따라 아이템렌더러로 개별 행들을 만들어 컬럼헤더 밑에 붙여 나가게 되는 것입니다.

5.2. 용어 설명

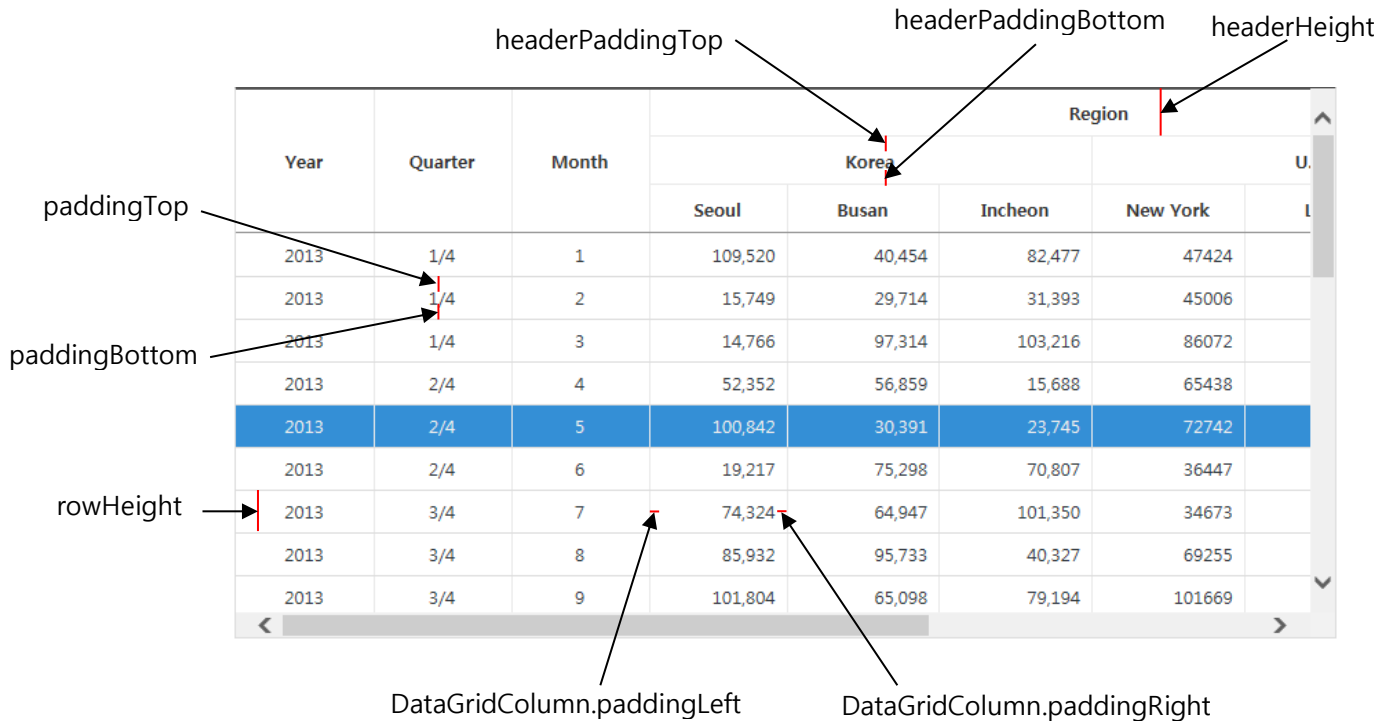
- 그리드(DataGrid)
  그리드 또는 데이터그리드로 행(Row, 로우), 열(Column, 컬럼)을 표시하는 컴포넌트.
  레이아웃에서 <DataGrid> 노드에서 정의하여 생겨나는 그리드 객체로 rMate Grid 의 가장 중심 컴포넌트입니다.
- 컬럼(column)

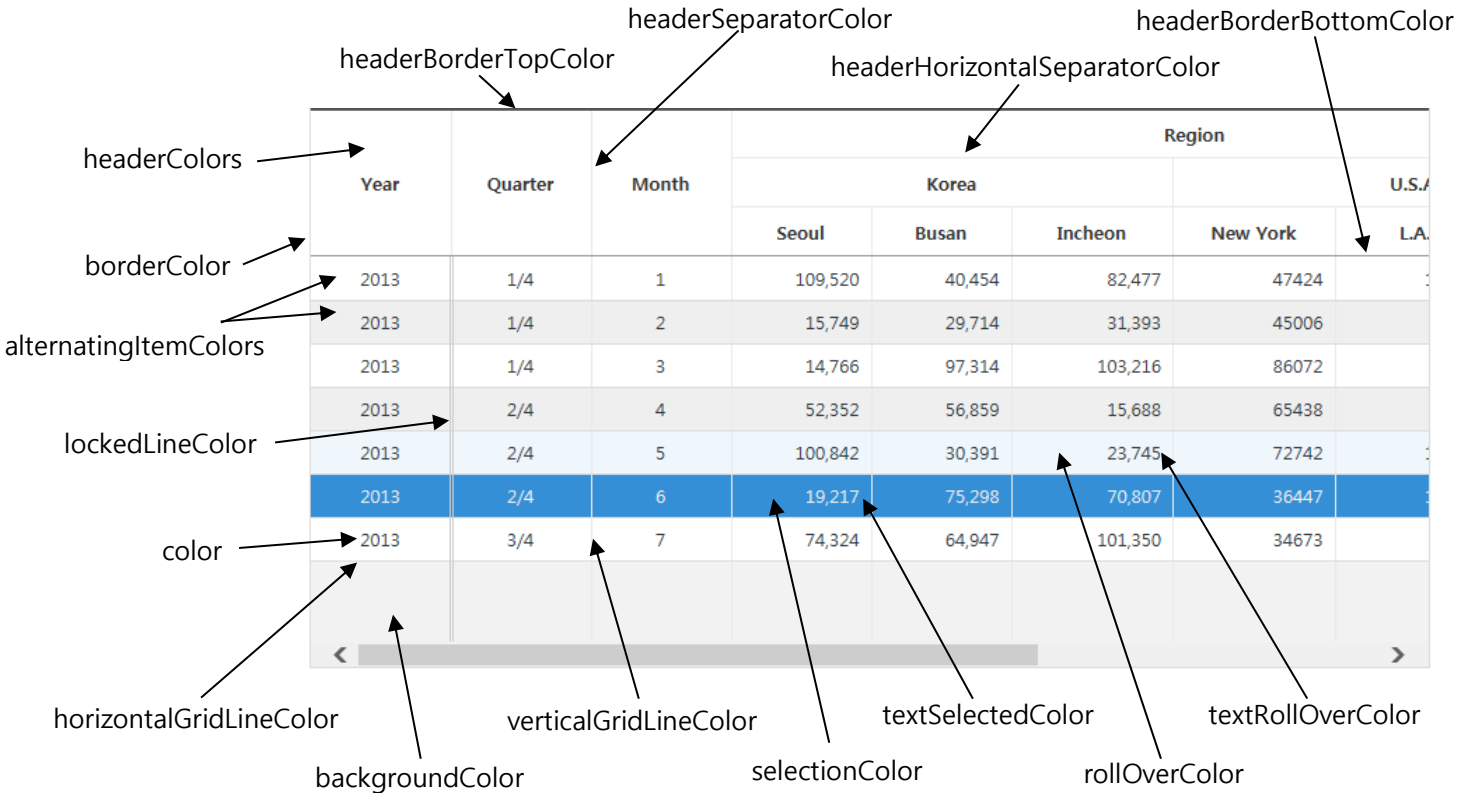
그리드의 개별 컬럼을 뜻합니다.
레이아웃에서 컬럼을 정의하는 것으로 <DataGridColumn>, <DataGridRowStateColum> 또는 <DataGridSelectorColumn>이 있으며, 각 컴포넌트는 실제 컬럼의 내용을 포함하거나 작업하는 것은 아니고 컬럼에 대한 정보만을 가지고 있게 됩니다.

- 컬럼그룹(ColumnGroup)
  개별 컬럼을 모아 계층구조를 나타내기 위한 컬럼
  실제 컬럼과는 달리 하위에 반드시 컬럼을 포함해야 하며 헤더를 표시하는 속성과 관련된 기능을 가집니다.
  레이아웃에서 <DataGridColumnGroup>노드로 정의하며, <DataGrid>의 groupedColumns 속성에 지정해 줍니다. (컬럼그룹이 없이 컬럼들만 있는데 groupedColumns 에 설정할 경우 성능에 영향을 주므로 컬럼만 있을 경우에는 columns 속성에 지정합니다)
- 헤더(Header)
  그리드에서 맨위의 컬럼정보를 나타내는 부분을 말합니다.
- 헤더 렌더러(headerRenderer)
  그리드에서 헤더를 표현하기 위해 생성하는 컴포넌트로 해당 컬럼의 headerRenderer 속성에서 사용할 컴포넌트명을 명시하며 그리드에 의해 생성 관리됩니다.
  headerRenderer 에 명시를 안하면 일반 텍스트가 표시되는 헤더 렌더러가 사용되고, 그외에 ComboBoxHeader, CheckBoxHeader 가 있습니다.
- 아이템 렌더러(itemRenderer)
  그리드에서 셀을 표현하기 위해 생성하는 컴포넌트로 해당 컬럼의 itemRenderer 속성에서 사용할 컴포넌트명을 명시하며, 그리드에 의해 생성 관리됩니다.
  itemRenderer 에 명시를 안하면 일반 텍스트가 표시되는 아이템 렌더러가 사용되고, 그외에는 HtmlItem, HtmlEditableItem, IconItem, ImageItem 등등이 있습니다.
- 아이템 에디터(itemEditor)
  그리드에서 셀을 편집하기 위해 생성하는 컴포넌트로 해당 컬럼의 itemEditor 속성에서 사용할 컴포넌트명을 명시하며, 그리드에 의해 생성 관리됩니다.
  itemEditor 에 명시를 안하면 일반 텍스트를 에디팅하는 아이템 에디터(TextInput)가 사용되고, 그외에는 CheckBoxEditor, ComboBoxEditor, ExComboBox, DateEditor, MonthEditor, NumericStepper, AutocompleteTextInput, TextArea 가 있습니다.
- 셀렉터(selector)
  그리드에서 행의 선택을 표현하기 위해 쓰이는 컴포넌트로 컬럼과 동일한 레벨에서 정의하고 사용되지만 차이점은 데이터의 특정 필드와 연결되지는 않는다는 점입니다.
- 콜렉션(collection)
  데이터를 저장하는 컴포넌트들로 배열과 XML 등을 담을 수 있습니다.
  사용자가 넣은 데이터는 rMate Grid 내에서 콜렉션 객체로 변환되는데 변환후에는 데이터에 대한 조작을 할 수 있는 함수 및 이벤트를 지원하는 기능을 제공하게 됩니다. 따라서 데이터에 조작은 반드시 콜렉션 객체를 통해 이루어져야 합니다.
  rMate Grid 에서 제공되는 콜렉션은 ArrayCollection, XMLListCollection, SpanArrayCollection,

SpanXMLListCollection, SpanSummaryCollection, HierachicalData, GroupingCollection, SpanHierachicalData, SpanGroupingCollection 이 있으며 레이아웃에서 dataProvider 에 특별히 콜렉션을 정의하지 않은 경우 데이터의 종류에 따라 배열이면 ArrayCollection, XML 이면 XMLListCollection 이 사용됩니다. 또한 병합과 같은 셀 속성이나 행속성을 제어하시려면 반드시 Span??? 류의 콜렉션을 지정해 줘야만 그리드에서 제어가 가능합니다.

5.3. 그리드 스타일 구조 설명





스타일명중 클래스명이 지정되지 않은 것은 모두 DataGrid 의 스타일 또는 속성입니다.

6. 그리드 설정 및 자바스크립트와의 연동

rMate Grid 의 기본 설정을 위하여 jsVars 를 사용합니다. 이의 사용법 및 기본 변수에 대해서 설명하겠습니다.

6.1. jsVars 설정

jsVars 변수는 rMate Grid 의 데이터, 레이아웃 등을 설정할 수 있는 가장 기본적인 변수입니다. 이 변수는 스크립트에서 그리드를 표시할 때 파라메터로 함께 삽입하게 됩니다.

jsVars 변수 설정은 다음과 같은 규칙을 따릅니다.

- jsVars 변수는 문자열 또는 object 입니다.
- 문자열일 경우 query string 의 형식을 따르며 두 개 이상의 설정을 하기 위해 구분자 & 를 사용합니다. 예를 들어 데이터 XML URL 과 레이아웃 XML URL 을 설정하는 방법은 아래와 같습니다

```
var layoutURL = "./Layout.xml";
var jsVars = "layoutURL="+layoutURL;

var dataURL = "./singleData.xml";
jsVars += "&dataURL="+dataURL;
```

<예제 12 문자열 jsVars 적용 예>

- object 일 경우에 데이터 XML URL 과 레이아웃 XML URL 을 설정하는 방법은 아래와 같습니다

```
var jsVars = {
    "layoutURL" : "./Layout.xml",
    "dataURL" : "./singleData.xml"
};
```

<예제 13 object jsVars 적용 예>

jsVars 로 설정 가능한 속성값은 다음과 같습니다.

1. 그리드 레이아웃 XML URL 경로
2. 그리드 데이터 XML URL 경로
3. rMateOnLoadCallFunction: 그리드 준비가 완료된 경우 호출할 함수 설정 (최초 1 번만 호출됨)
4. assetsPath: Icon 이미지의 경로 (그리드에 입력되는 icon 이미지와 iconRenderer 에 포함되는 이미지의 경로입니다. 또한, 자바스크립트에서 rMateGridH5.setAssetsPath("경로")와 같은 함수를 이용하여 이미지 경로를 변경할 수 있습니다.)

5. 읽어들이는 데이터의 유형이 XML 형식일 경우

```
jsVars += "&dataType=xml";
```

6. 다국어 설정 (그리드내에서 메시지 표현시 사용할 언어 설정)

```
jsVars += "&localeChain=en-US";
```

7. 웹접근성 지원 설정(스크린 리더기가 읽을 문자열을 그리드에 넣음)

```
jsVars += "&accessibility=true";
```

위와 같이 jsVars 를 설정한 후 실제적으로 그리드를 생성하는 함수에 설정한 jsVars 를 파라메터로 넣어 줍니다.

rMateGridH5.create ("grid1","gridHolder", jsVars, "800px", "600px");

이때 rMateGridH5.create 함수의 파라메터는 다음과 같습니다. (순서대로)

1. ID (임의로 정의하십시오)
2. 그리드가 표현될 div의 ID
3. 초기 실행함수나 URL 경로 등을 정의한 jsVars변수.
4. rMateGrid 가로 사이즈.
5. rMateGrid 세로 사이즈.

6.2. 그리드 연동을 위한 기본 설정 및 변수

자바스크립트 연동을 위해 기본적으로 사용하는 변수와 사용 예를 보면 다음과 같습니다.

```
<!DOCTYPE html>
<html>
<title>rMateGridH5 (RiaMore Soft)</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="Content-Script-Type" content="text/javascript" />
<meta http-equiv="Content-Style-Type" content="text/css" />

<link rel="stylesheet" type="text/css" href="./rMateGridH5Sample.css"/>

<link rel="stylesheet" type="text/css" href="../rMateGridH5/Assets/rMateH5.css"/>
<!-- rMateGridH5 라이센스 -->
<script type="text/javascript"
src="../LicenseKey/rMateGridH5License.js"></script>
<!-- rMateGridH5 라이브러리 -->
<script type="text/javascript" src="../rMateGridH5/JS/rMateGridH5.js"></script>

<!-- 사용자 정의 설정 시작 -->
<script language="JavaScript" type="text/javascript">

// ---------------------- rMateGrid jsVars 설정 시작----------------------
```

```
// rMateGrid 레이아웃 URL 경로. 반드시 설정하십시오.
var layoutURL = "LayoutSimple.xml";
var jsVars = "layoutURL="+layoutURL;

// 데이터를 URL 경로를 통해 가져올 경우 설정하십시오.
// 배열형태로 데이터를 삽입할 경우 주석처리나 삭제하십시오.
// 배열형태와 같이 사용할 경우, 우선순위는 배열형태 데이터 삽입입니다.
var dataURL = "./DataXml/DataOneDepthRevenues.xml";
jsVars += "&dataURL="+dataURL;

// ---------------------- rMateGrid jsVars 설정 끝 ----------------------


// rMateGrid 관련 객체
var gridApp;  // 그리드 rMateGridH5 기초 div                          1
var gridRoot; // 데이터와 그리드를 포함하는 객체
var dataGrid; // 그리드

// rMateGrid가 로딩된 후 불려지는 함수
var gridReadyHandler = function(id) {                                 3
	gridApp = document.getElementById(id);
	gridRoot = gridApp.getGridRoot();

	var layoutCompleteHandler = function(event) {
		dataGrid = gridApp.getDataGrid();                             4
	}

	gridRoot.addEventListener("layoutComplete", layoutCompleteHandler);
}

// rMateGridH5에서 그리드 생성 준비가 완료될 경우 호출할 함수를 지정합니다.
jsVars += "&rMateOnLoadCallFunction=gridReadyHandler";                2

rMateGridH5.create("grid1", "gridHolder", jsVars, "100%", "100%");


</script>
<!-- 사용자 정의 설정 끝  -->
</head>
<body>
	<div class="content">
		<!-- 그리드가 삽입될 DIV -->
		<div id="gridHolder">
		</div>
	</div>
</body>
</html>
```

<예제 14 자바스크립트와 연동 변수 예>

1. 자바스크립트를 통해 rMate Grid 에서 접근하는 기본 객체로 3 개를 선언하였습니다.
   - gridApp : 그리드 rMateGridH5 기초 div
   - gridRoot : 그리드와 데이터를 포함하는 객체로 레이아웃과 데이터를 설정한 내용에 따라 로딩하고 그리드를 생성하는 객체입니다. 그리드가 로딩된 후 데이터를 읽어 들이게 되면

"dataComplete" 이벤트가 발생하고 layout 에 의해 그리드가 만들어 지면 "layoutComplete" 이벤트가 발생하게 됩니다.

- dataGrid : 설정된 레이아웃에 따라 그리드가 생성되면 넣어지는 변수

2. rMate Grid 에서 그리드를 만들 준비를 마친 후 rMate Grid 가 호출할 함수를 설정합니다. 위의 예에서는 로딩 완료 후 `gridReadyHandler` 함수를 호출하도록 지정하였습니다.
3. `gridReadyHandler` 함수에서 1 번에서 선언한 변수에 객체를 세팅합니다.
4. dataGrid 변수는 로딩과 동시에 만들어지는 것이 아니므로 이벤트를 통해 그리드가 만들진 후에 세팅되도록 합니다. 또한 생성된 객체들에 이벤트를 걸어야 하는 경우 해당 객체에 적합한 이벤트를 찾아 연동할 함수와 함께 `addEventListener` 를 통하여 지정하도록 합니다. 객체별로 발생하는 이벤트는 API Reference 문서에서 이벤트 부분을 찾아 보시면 됩니다.

6.3. rMate Grid 컴포넌트의 속성 연동

rMate Grid 의 컴포넌트에 접근할 때 컴포넌트의 속성값(Property)을 지정하거나 조회하는 방법을 설명하겠습니다.
자바스크립트에서 그리드 객체 속성에 접근하는 방법은 속성에 따라 직접 접근하는 방법(컴포넌트 설명의 속성 및 유효 값 설명)과 getter, setter 함수(컴포넌트 설명의 속성 함수명(getter/setter) 설명)를 통해서 접근하는 방식을 사용하여야 하며 어느방식으로 접근해야하는지는 객체의 속성 에 대한 설명을 참조해야 합니다.

```
var selectionChangeHandler = function(event) {
	var rowIndex = event.rowIndex;
	var columnIndex = event.columnIndex;
	var dataRow = gridRoot.getItemAt(rowIndex);
	inputForm.itemXMLData.value = dataRow;
	inputForm.rowIndex.value = rowIndex;
	var column = dataGrid.getColumns()[columnIndex];
	var dataField = column.getDataField();
}

function selectedUp() {
	inputForm.selectedIndex.value = inputForm.selectedIndex.value - 1;
	dataGrid.setSelectedIndex(inputForm.selectedIndex.value);
}
```

<예제 15 컴포넌트 속성 연동 예>

event 의 columnIndex 는 속성에 직접 접근하여 값을 가져오는 방식을 사용하며, dataGrid.getColumns() 는 dataGrid 객체의 columns 속성을 가져오라는 뜻이 되며, 반환 객체가 배열이므로 배열에서 columnIndex 번째 객체를 가져오는 문장입니다.
세번째 예는 dataGrid 의 selectedIndex 속성에 값을 지정하는 예로 setSelectedIndex()함수를 사용하여 값을 지정하고 있습니다.

6.4. rMate Grid 컴포넌트의 스타일 연동

rMate Grid 에서 스타일을 설정할 수 있는 방법은 레이아웃에서 개별 컴포넌트 선언시 스타일을 지정하는 방법과 <Style>에서 CSS 형식으로 스타일을 만들고 해당 CSS 를 원하는 컴포넌트에 styleName 에 의거하여 지정하는 방법, 자바스크립트를 통해 setStyle 함수를 사용하여 지정하는 방법이 있습니다.
레이아웃에 의거한 스타일 설정방법은 4.5 레이아웃의 Style 노드(CSS 적용) 설정하기 에서 설명하였으므로, 여기서는 자바스크립트를 이용하는 방법에 대해 설명하겠습니다.
자바스크립트에서 그리드 객체의 스타일을 지정하거나 조회하는 경우에는 객체의 setStyle, getStyle 함수(px 등이 붙는 스타일에 대해 숫자로 가져오려면 setPxStyle 과 getSizeStyle 을 사용)를 사용합니다.

```
function changeStyle() {
	var color = dataGrid.getStyle("color");
	dataGrid.setStyle("color", "#FF0000");
}

function changePxStyle() {
	var paddingTop = dataGrid.getSizeStyle("paddingTop");
	dataGrid.setPxStyle("paddingTop", paddingTop + 1);
}
```

<예제 16 컴포넌트 스타일 연동 예>

위의 예에서는 dataGrid 의 스타일 속성인 color 의 값을 가져와 #FF0000 로 설정하는 예와 paddingTop 을 가져와서 1 을 더해 다시 지정하는 예 입니다.
getSizeStyle 함수를 통해 paddingTop 의 값을 가져와서 setPxStyle 함수를 사용하여 값을 지정하고 있습니다.
여기서 주의할 점은 모든 객체가 스타일을 가지고 있는 것은 아니며, 스타일 속성이 무엇이 있는지를 확인해야 한다는 점입니다. 이를 확인하기 위해서는 API Reference 문서에서 원하는 객체의 함수에 getStyle, setStyle 함수의 존재 여부를 확인하고 조정할 수 있는 스타일을 찾아 조정하는 작업이 필요합니다.
만일 레이아웃에 스타일을 미리 지정해 놓았고 자바스크립트에서 해당 스타일을 적용하고자 한다면 styleName 속성을 이용하여 객체.setStyleName(스타일명) 함수를 사용하여 지정하면 됩니다.(단 해당 컴포넌트가 styleName 속성이 있어야 합니다)

6.5. rMate Grid 컴포넌트의 이벤트 연동

rMate Grid 내의 컴포넌트에 대해 이벤트 발생시에 자바스크립트 함수를 호출할 수 있도록 하여 이벤트 연동을 할 수 있습니다.

```
// rMateGrid가 로딩된 후 불려지는 함수
var rMateGridOnLoad = function(id) {
	gridApp = document.getElementById(id);  // id는 rMateCreate 시 설정한 Grid의 ID
	gridRoot = gridApp.getGridRoot();

	var selectionChangeHandler = function(event) {
		var rowIndex = event.rowIndex;
		var columnIndex = event.columnIndex;
		inputForm.selectedIndex.value = rowIndex;
	}

	var layoutCompleteHandler = function(event) {
		dataGrid = gridApp.getDataGrid();
		dataGrid.addEventListener("change", selectionChangeHandler);
		// 미리 선택된 행을 설정
		dataGrid.setSelectedIndex(6);
	}

	gridRoot.addEventListener("layoutComplete", layoutCompleteHandler);
}
```

<예제 17 컴포넌트 이벤트 연동 예>

위의 예에서는 dataGrid 에서 발생하는 "change" 이벤트(사용자가 마우스를 클릭하여 행이나 셀을 선택한 경우에 발생)에 대해 selectionChangeHandler 함수를 호출하도록 설정한 경우입니다.
이벤트가 발생하여 selectionChangeHandler 함수가 호출되면 발생된 이벤트의 내용이 event 파라메터로 전달되고 event 변수에서 상세한 내용을 받아 볼 수 있습니다.
rMate Grid 의 컴포넌트에서 지원되는 Event 는 크게 두 종류로 HTML 에서 발생하는 이벤트와 그리드 시스템이 자체적으로 발생시키는 이벤트입니다. 이때 이벤트 처리 함수를 등록할 때에는 이벤트의 종류에 따라 HTML 이벤트는 addEvent, removeEvent 를, 그리드 자체 이벤트는 addEventListener, removeEventListener 를 사용하셔야 합니다.
rMate Grid 의 컴포넌트중 이벤트를 발행하는 객체와 객체별로 발생되는 이벤트의 종류, 이벤트 종류에 따른 이벤트 상세내역은 API Reference 문서에서 자세히 살펴보실 수 있으며, 컴포넌트에 addEventListener, removeEventListener, addEvent, removeEvent 등의 함수가 없는 컴포넌트는 이벤트를 받을 수 없는 객체입니다.

7. 에디팅

rMate Grid 에서 데이터를 에디팅하는 방법은 크게 2 종류로 자바스크립트를 이용하는 방법과 그리드내의 편집기능을 이용하여 수정하는 방법입니다.
(그리드의 데이터가 그룹핑일 경우에는 에디팅은 지원되지 않습니다)

7.1. 자바스크립트를 통한 데이터 수정

자바스크립트를 통해 데이터를 수정하기 위해 자료를 수정하는 예는 다음과 같습니다.

```
// rMateGrid 관련 객체
var gridApp;      // 그리드 rMateGridH5 기초 div
var gridRoot;     // 데이터와 그리드를 포함하는 객체
var dataGrid;     // 그리드

// rMateGrid가 로딩된 후 불려지는 함수
var rMateGridOnLoad = function(id) {
        gridApp = document.getElementById(id);
        gridRoot = gridApp.getGridRoot();

        var selectionChangeHandler = function(event) {
                var rowIndex = event.rowIndex;
                var columnIndex = event.columnIndex;
                var dataRow = gridRoot.getItemAt(rowIndex);
                inputForm.itemXMLData.value = dataRow;
                inputForm.rowIndex.value = rowIndex;
                var column = dataGrid.getColumns()[columnIndex];
                var dataField = column.getDataField();
        }

        var layoutCompleteHandler = function(event) {
                dataGrid = gridApp.getDataGrid();
                dataGrid.addEventListener("change", selectionChangeHandler);
        }

        gridRoot.addEventListener("layoutComplete", layoutCompleteHandler);
}

// 전체 삭제
function removeAll() {
        gridRoot.removeAll();
}
```

```
// 행 삭제
function removeItemAt() {
	// 현재 선택된 행을 삭제 합니다.
	gridRoot.removeItemAt();
	// 원하는 행의 번호를 통해 행을 삭제합니다.
//	gridRoot.removeItemAt(inputForm.rowIndex.value);
}

// 행 수정
function setItemAt() {
	var item = inputForm.itemXMLData.value;
	gridRoot.setItemAt(item, inputForm.rowIndex.value);
}

// 행 추가
function addItemAt() {
	var item = inputForm.itemXMLData.value;
	gridRoot.addItemAt(item, inputForm.rowIndex.value);
}

// 맨끝에 행 추가
function addItem() {
	var item = inputForm.itemXMLData.value;
	gridRoot.addItemAt(item);
}

// 수정 작업 결과 가져오기 (그리드에서 작업된 입력,수정,삭제 내용을 가져옵니다)
// 데이터는 배열 형태로
//	idx: 행번호
//	job: 수행 작업 (I:입력, U:수정, D:삭제)
//	data: 행의 자료
// 를 가지고 있으며 삭제가 먼저 오도록 정렬되어 있습니다.
function getChangedData() {
	var changedData = gridRoot.getChangedData();
	if (changedData.length == 0)
		alert("변경된 자료가 없습니다");
	else {
		for (var i = 0; i < changedData.length; i++)

	alert("index:"+changedData[i].idx+"₩n"+"job:"+changedData[i].job+"₩n"+"data:"+changedData[i].
```

```
data);
        }
}

// 수정 작업 결과 XML로 가져오기 (그리드에서 작업된 입력,수정,삭제 내용을 XML로 가져옵니다)
// 데이터는 XML 형태로
//        idx: 행번호
//        job: 수행 작업 (I:입력, U:수정, D:삭제)
//        data: 행의 자료
// node를 가지고 있으며 삭제가 먼저 오도록 정렬되어 있습니다.
function getChangedDataXML() {
        var changedDataXML = gridRoot.getChangedDataXML();
        if (changedDataXML == null)
                alert("변경된 자료가 없습니다");
        else
                alert(changedDataXML);
}
```

<예제 18 자바스크립트를 이용한 데이터 수정 예>

위의 예에서는 사용자가 그리드에서 한 개의 행을 선택하면 자바스크립에서 선택된 행에 대한 정보(행번호등)을 받아 선택된 행의 data 를 gridRoot.getItemAt 함수를 호출하여 읽어오고, 이후 사용자의 수정이 진행된후 gridRoot.setItemAt 함수를 사용하여 그리드의 data 를 수정하거나, gridRoot.removeItemAt 함수를 사용하여 해당 행을 삭제하는 기능을 설명하였습니다.
또한 전체삭제는 gridRoot.removeAll 함수를 호출하여 작업할 수 있으며, 새로운 행을 추가하는 경우에는 gridRoot.addItemAt 함수를 사용하여 원하는 행 또는 맨 마지막 행에 새로운 data(행의 index 번호 파라메터를 안주거나 -1 을 세팅)를 등록할 수 있습니다.

매번 변경사항 생길 때마다 서버와의 작업으로 서버의 데이터도 변경하는 것도 가능하나, 사용자가 변경작업을 모두 처리하고 한꺼번에 수정사항을 모두 반영하는 경우를 위해, 그리드에서는 위의 함수 호출에 대해 자료를 저장하는 기능이 제공됩니다.
이런 기능을 제공하기 위해 gridRoot.getChangedData, gridRoot.getChangedDataXML 함수를 제공하는데 데이터가 배열일 경우와 XML 일 경우에 따라 불러 쓰면 되며 getChangedData 함수의 반환값은 다음 형식의 배열입니다.

idx : 행번호
job : 수행된 작업을 나타내며 다음 값을 가집니다.
I:등록, D:삭제, U:수정
data : 사용자가 등록 또는 수정한 내용(행의 모든 데이터가 저장됨)

getChangedDataXML 함수의 반환값은 위 데이터를 XML 로 표현하여 문자열로 반환하게 됩니다.

이때 배열의 순서는 사용자의 에디팅 순서이나 서버작업의 편의를 위해 job 이 "D"인 것 즉 삭제된 자료가 맨 먼저 나타나도록 하였습니다.

자바스크립트를 이용하여 행을 수정하는 방법은 그리드의 데이터가 배열이냐 XML 이냐에 따라 약간 달라지는데, 이유는 개별 행의 자료를 넘겨 받아 처리하는 방식에서 차이가 나기 때문입니다. 배열의 경우 전달받은 행의 data 는 자바스크립트에서 직접인식이 가능한 객체이기 때문에 일반적인 자바스크립트 코딩방식을 써서 조작이 쉬운 반면, XML 로 행의 data 를 받게되면 XMLElement 를 받은 것이기 때문에 데이터를 조작하기 위해서는 XML 에 대한 조작 방식에 따른 작업이 필요하게 됩니다. XML 의 처리과정의 예는 샘플에서 "XML 데이터 제어" 메뉴에서 간략히 코딩되어있으니 참고하시기 바랍니다.

행을 수정하는 방법외에 개별 필드를 수정하는 방법으로 gridRoot.getItemFieldAt 와 gridRoot.setItemFieldAt 함수가 제공됩니다. 행의 수정이 XML 여부에 따라 바뀌는 것과 달리 데이터의 형식에 구애되지 않으므로 편리하며 사용 예는 샘플에서 "필드 데이터 제어" 메뉴를 참고하시기 바랍니다.

7.2. 그리드의 편집기능을 통한 데이터 수정

그리드의 편집기능을 이용하기 위해서는 레이아웃을 구성할 때 다음과 같은 속성을 구성합니다.

```
<rMateGrid>
	<DataGrid id="dg1" editable="true" doubleClickEnabled="true"
		itemEditBeginningJsFunction="itemEditBeginningChecker"
itemEditEndJsFunction="itemEditEndChecker"
		selectionMode="singleRow">
		<columns>
			<DataGridColumn dataField="Region" editable="false"/>
			<DataGridColumn dataField="Territory" width="150"/>
			<DataGridColumn dataField="Territory_Rep"
headerText="Territory Rep" width="150"/>
			<DataGridColumn dataField="Actual"/>
			<DataGridColumn dataField="Estimate"/>
			<DataGridColumn dataField="Real"/>
			<DataGridColumn dataField="Price"/>
		</columns>
	</DataGrid>
</rMateGrid>
```

<예제 19 편집용 레이아웃 예>

위의 예에서 editable 속성은 그리드의 수정모드로 반드시 true 를 주어야하며, doubleClickEnabled 는 더블클릭 가능여부로 더블클릭에 의거해서 셀 편집모드로 전환할 지를 지정할 수 있습니다.
itemEditBeginningJsFunction 는 자바스크립트 함수를 통해 해당 셀의 수정가능여부를 판별하기 위한 기능으로 사용자 특정 셀을 클릭 또는 더블클릭으로 수정 모드로 들어가려 하면 itemEditBeginningJsFunction 에서 지정한 자바스크립 함수가 수행되어 결과가 true 일 경우에는 셀 수정

모드로 전환되고, false 일 경우에는 들어가지지 않습니다. (CheckBoxItem 이나 ComboBoxItem 과 같이 이미 수정이 가능한 상태의 렌더러에서는 제대로 작동하지 않습니다)
itemEditEndJsFunction 는 자바스크립트 함수를 통해 셀의 수정 값의 반영여부를 판별하는 기능으로, 사용자가 셀의 수정을 마치게 되면 itemEditEndJsFunction 에서 지정한 자바스크립트 함수가 수행되어 정상적인 값이 입력되었다면 null, 오류가 있을 경우에는 오류메시지를 반환하게 하여 수정내용의 반영여부를 조작할 수 있게 됩니다. 이때 표시되는 오류메시지는 아래 그림과 같습니다.

Selected	From	Subject	ReceiveDate	CC	Length	AttachCount
No	Amit	GMC-1 Release1	2013/12/07	Jim	75,126	1
No	Amit	GMC-1 Release2	2013/12/07	Jim	95,822	2
No	Amit	GMC-1 Release3	2013/12/11	Jim	1,862,456	0
No	Barb	GMC-1 Release3	2013/12/08	Anant	65,488	31
No	Barb	GMC-1 Release4	2013/12/06	Anant	4,851,568	1
Yes	Harry	GMC-1 Release2	2013/11/28	Ram	13,548	0
No	Harry	GMC-1 Release3	2013/12/08	Ram	1	Length값은 1000이상으로 입력하셔야 합니다.
No	John	GMC-1 Release2	2013/12/07	Jim	84,621,877	20
No	John	GMC-1 Release3	2013/12/07	Jim	51,846	3
No	Phill	GMC-1 Release1	2013/12/08	Jim	31,680	5
No	Phill	GMC-1 Release1	2013/12/08	Jim	1,813	0
No	Phill	GMC-1 Release3	2013/12/08	Jim	13,548	1

<그림 6 그리드 편집 오류 메시지표시>

또한 전체 그리드에 대해 수정모드로 하였어도 개별 컬럼에 대해 editable 속성을 주어 수정이 안되도록 지정하실 수 있습니다.

다음은 그리드에서 편집을 위해 위해 제공되는 컴포넌트(아이템에디터)를 설명하겠습니다.
그리드에서 제공되는 아이템에디터에는 텍스트, 체크박스, 콤보박스, 날짜선택창, TextArea 등이 있으며 기본으로는 텍스트(TextInput)가 사용되고 필요시 지정할 수 있으며 레이아웃에서 다음과 같이 지정하시면 됩니다.

```
<rMateGrid>
	<DataGrid id="dg1" editable="true" doubleClickEnabled="true"
verticalAlign="middle">
		<columns>
			<DataGridColumn dataField="Selected" width="80"
itemEditor="CheckBoxEditor" editorDataField="selected"
labelJsFunction="convertYesNo"/>
			<DataGridColumn dataField="From" width="80"/>
			<DataGridColumn dataField="Subject" width="150"
itemEditor="ComboBoxEditor"
				itemRendererDataProvider="[{'label':'GMC-1
```

```
Release1'},{'label':'GMC-1 Release2'},{'label':'GMC-1 Release3'},{'label':'GMC-1
Release4'},{'label':'GMC-1 Release5'},{'label':'GMC-1 Release6'}]"/>
                    <DataGridColumn dataField="Date" itemEditor="DateEditor"/>
                    <DataGridColumn dataField="CC"/>
                    <DataGridColumn dataField="Profit" textAlign="right"
formatter="{numfmt}"/>
                </columns>
        </DataGrid>
</rMateGrid>
```

<예제 20 편집용 컴포넌트 사용 예>

아이템에디터를 지정하지 않으면 일반 문자열을 넣을 수 있는 기능의 텍스트 아이템에디터가 사용되며 itemEditor 속성에 원하는 아이템에디터의 이름을 넣어주면 됩니다.
아이템에디터를 CheckBoxEditor 를 사용하실 경우에는 반드시 editorDataField 속성에 selected 를 넣어주셔야 합니다.
ComboBoxEditor 를 사용하실 경우, 콤보박스에 고정된 내용을 보여주는 경우 itemRendererDataProvider 속성을 이용하여 콤보박스의 데이터를 JSON 형식으로 넣어주면 되나, 행마다 내용이 바뀌는 경우에는 itemRendererDataProviderField 속성을 이용하여 데이터에 넣어진 특정 필드에서 데이터를 가져오도록 설정할 수 있습니다.

8. 행, 셀 속성 제어

rMate Grid 에서 행에 대해서 줄 수 있는 속성은 배경색, 스타일(폰트관련이나 정렬등), 포맷, 잠금(에디팅 불가), 행의 높이이며, 셀에 대해서 줄 수 있는 속성은 배경색 스타일, 포맷, 병합정보이며, 행, 셀에 속성을 지정하는 방법은 함수를 이용하는 방법과 데이터에 속성정보를 넣어서 표현하는 방법이 있습니다.
주의할 점은 행, 셀속성을 지정할 수 있으려면 그리드에 주는 데이터의 형식이 행, 셀속성을 넣을 수 있는 콜렉션(Span????로 시작되는 콜렉션)이어야 한다는 것입니다. 그리드 자체에는 행, 셀의 속성을 저장하는 기능은 없으며, 관련정보는 데이터를 포함하는 콜렉션에서 저장, 관리됩니다. rMate Grid 에서 이를 지원하는 콜렉션은 SpanArrayCollection, SpanXMLListCollection, SpanSummaryCollection, SpanHierachicalData, SpanGroupingCollection 이 있습니다.

8.1. 행 속성

행 속성을 저장하는 컴포넌트는 SpanRowAttribute 로 행에 줄수 있는 속성은 다음과 같습니다.

속성 명	유효 값(기본값)	설명
backgroundColor	RGB	행의 배경색 #FFFFFF 형식으로 표현합니다.
editable	Boolean(true)	행 수정 가능여부

		그리드의 editable속성에 의한 편집 모드에서 클릭이나 더블클릭시에도 에디팅모드로 변환되지 않도록 합니다.
formatString	String	마스크 패턴 패턴적용이 실패하는 경우에는 원래의 데이터가 그대로 표시됩니다. 자세한 내역은 API Reference 문서의 SpanRowAttribute를 참조하세요
rowHeight	Number	행의 높이(픽셀단위). DataGrid의 variableRowHeight가 true일 때 작동합니다.
styleName	String	행에 적용할 스타일명 레이아웃의 <Style>노드에서 선언되거나 별도의 CSS파일에서 선언된 스타일명을 적어줍니다.

<표 2 행에 지정할 수 있는 속성>

8.2. 셀 속성

셀 속성을 저장하는 컴포넌트는 SpanCellAttribute 로 셀에 줄수 있는 속성은 다음과 같습니다.

속성 명	유효 값(기본값)	설명
backgroundColor	RGB	셀의 배경색 #FFFFFF 형식으로 표현합니다.
colNo	int	컬럼의 index번호
colSpan	int(1)	병합할 컬럼의 수
formatString	String	마스크 패턴 패턴적용이 실패하는 경우에는 원래의 데이터가 그대로 표시됩니다. 자세한 내역은 API Reference 문서의 SpanCellAttribute를 참조하세요
rowSpan	int(1)	병합할 행의 수
styleName	String	셀에 적용할 스타일명 레이아웃의 <Style>노드에서 선언되거나 별도의 CSS파일에서 선언된 스타일명을 적어줍니다.

<표 3 셀에 지정할 수 있는 속성>

8.3. 자바스크립트를 이용한 행, 셀 속성지정

자바스크립트에서 행, 셀에 속성을 지정하는 예는 다음과 같습니다.

```
// rMateGrid 관련 객체
var gridApp;      // 그리드 rMateGridH5 기초 div
var gridRoot;     // 데이터와 그리드를 포함하는 객체

// rMateGrid가 로딩된 후 불려지는 함수
var rMateGridOnLoad = function() {
        gridApp = document.getElementById("rMateCreate 시 설정한 Grid의 ID");
        gridRoot = gridApp.getGridRoot();

        var dataCompletedHandler = function(event) {
                setSpanAttributes();
        }

        gridRoot.addEventListener("dataComplete", dataCompletedHandler);
}

var collection;    // 그리드의 데이터 객체

function setSpanAttributes() {
        if (collection == null)
                collection = gridRoot.getCollection();
        if (collection == null) {
                alert("collection 객체를 찾을 수 없습니다");
                return;
        }

        collection.addRowAttributeDetailAt(4, "subTotalStyle", #DDDDDD, "#,##0원", true, 40);

        collection.addCellAttributeDetailAt(0, 0, 12, 1);
        collection.addCellAttributeDetailAt(12, 0, 12, 1, "subTotalStyle", #FFCC00);
        for (i = 0; i < collection.getLength(); i++) {
                if (i % 3 == 0)
                        collection.addCellAttributeDetailAt(i, 1, 3, 1);
        }
}
function clearRowAttr() {
```

```
        if (collection == null)
                collection = gridRoot.getCollection();
        collection.removeRowAttributeAt(4);
        collection.removeCellAttributeAt(0, 0);
}
function clearAllAttr() {
        if (collection == null)
                collection = gridRoot.getCollection();
        collection.removeAllAttribute();
}
```

<예제 20 자바스크립트에서 행, 셀 속성 적용 예>

위 예를 보면 맨 먼저 rMate Grid 에서 collection 컴포넌트의 인스턴스를 받아오고, 받아온 collection 객체에 addRowAttributeDetailAt 함수, addCellAttributeDetailAt 함수를 사용하여 원하는 곳에 속성을 지정합니다. 또한 지정된 속성의 삭제는 removeRowAttributeAt 함수와 removeCellAttributeAt 함수를 사용하며 각 함수에 대한 설명은 다음과 같습니다.

행 속성을 등록하는 addRowAttributeDetailAt 함수의 파라메터는 다음과 같습니다.

function addRowAttributeDetailAt(rowNo:int, styleName:String, backgroundColor:RGB, formatString:String, editable:Boolean, rowHeight:Number)

- rowNo : 행의 index 번호
- styleName : 적용할 style 명 (layout 에서 정의한 스타일명('.'은 제외한 부분)
- backgroundColor : 배경색 (RGB 값으로 적용)
- formatString : 행에 적용할 formatString (매뉴얼 참조)
- editable : 행의 수정가능여부 (true/false)
- rowHeight : 행의 높이 (단위:pixel)

셀 속성을 등록하는 addCellAttributeDetailAt 함수의 파라메터는 다음과 같습니다.

function addCellAttributeDetailAt(rowNo:int, colNo:int, rowSpan:int, colSpan:int, styleName:String, backgroundColor:RGB, formatString:String)

- rowNo : 행의 index 번호
- colNo : 컬럼의 index 번호
- rowSpan : 병합할 행의 갯수 (기본 1 개)
- colSpan : 병합할 컬럼의 갯수 (기본 1 개)
- styleName : 적용할 style 명 (layout 에서 정의한 스타일명('.'은 제외한 부분)
- backgroundColor : 배경색 (RGB 값으로 적용)
- formatString : 셀에 적용할 formatString (매뉴얼 참조)

행 속성을 삭제하는 removeRowAttributeAt 함수의 파라메터는 다음과 같습니다.
function removeRowAttributeAt(rowNo:int)
rowNo : 행의 index 번호

셀 속성을 삭제하는 removeCellAttributeAt 함수의 파라메터는 다음과 같습니다.
function removeCellAttributeAt(rowNo:int, colNo:int)
rowNo : 행의 index 번호
colNo : 컬럼의 index 번호

모든 속성을 삭제하는 removeAllAttribute 함수는 다음 구조로 되어 있습니다.
function removeAllAttribute():Boolean

행, 셀의 속성을 데이터에 넣지 않고 모두 자바스크립트 함수로 설정하는 경우, 데이터에서 속성을 추출하는 작업이 필요없기 때문에 SpanArrayCollection, SpanXMLListCollection 콜렉션에는 extractable 속성이 있으며 이 extractable 속성을 false 로 하여 추출작업을 안하도록 하여야 합니다.

8.4. 데이터에 행, 셀 속성지정

rMate Grid 에서는 데이터에 직접 행 또는 셀의 속성을 지정하는 방식을 지원합니다.
개별적인 함수에 의한 제어가 아닌 서버측에서 데이터를 가공할 때 표현속성을 넣어 줌으로써 작업의 편의를 도모하고, 특히 데이터가 XML 형식일 때 마치 html 에서 table tag 에 속성을 넣어 제어하는 방식과 유사한 방식으로 제어할 수 있는 것입니다. 데이터가 XML 일 경우의 예를 들면 다음과 같습니다.

```
<?xml version="1.0" encoding="utf-8"?>
<RevenuesAnalysis>
    <RegionalRevenue>
        <Year>2007</Year>
        <Quarter>4/4</Quarter>
        <Month>12</Month>
        <Seoul>99008</Seoul>
        <Pusan>30598</Pusan>
        <Incheon>99124</Incheon>
        <NewYork>22776</NewYork>
        <LA>107805</LA>
        <Washington>38384</Washington>
        <Revenue>397696</Revenue>
        <Percent>35</Percent>
```

```
        </RegionalRevenue>

        <RegionalRevenue styleName="allTotalStyle" backgroundColor="#FFFF00" editable="false"
formatString="#,##0원" rowHeigh="50">
                <Year rowSpan="2" colSpan="3" styleName="allTotalHeaderStyle"
backgroundColor="#DDDDDD" formatString="#,##0">T O T A L</Year>
                <Seoul>846568</Seoul>
                <Pusan>748824</Pusan>
                <Incheon>717834</Incheon>
                <NewYork>676387</NewYork>
                <LA>780781</LA>
                <Washington>857351</Washington>
                <Revenue>4627745</Revenue>
                <Percent></Percent>
        </RegionalRevenue>
                :
                :
</RevenuesAnalysis>
```

<예제 21 XML 데이터에 행, 셀 속성 넣은 예>

```
<rMateGrid>
        <DataGrid id="dg1" editable="true" doubleClickEnabled="true"
verticalAlign="middle">
                <columns>
                        <DataGridColumn dataField="Year" width="80"/>
                        <DataGridColumn dataField="Quater" width="150"/>
                        <DataGridColumn dataField="NewYork"/>
                        <DataGridColumn dataField="Revenue"/>
                        <DataGridColumn dataField="Percent" textAlign="right"/>
                </columns>
                <dataProvider>
                        <!--
                         그리드의 자료를 일반 data형식(배열, XML)이 아닌 별도의 컴포넌트에
입력해야 할 경우 아래와 같이 $gridData를 넣어줍니다
                        -->
                        <SpanXMLListCollection source="{$gridData}"/>
                </dataProvider>
        </DataGrid>
</rMateGrid>
```

<예제 22 SpanXMLListCollection 컴포넌트 사용 예>

이 작업에 사용되는 컴포넌트는 SpanXMLListCollection 으로 행을 나타내는 RegionalRevenue 노드에서 행의 속성을 XML attribute 에 넣음으로써 간단하게 행 속성을 넣어 줄 수 있고, 셀을 나타내는 Year 노드에

서 셀의 속성을 XML attribute 에 넣어주게 됩니다. 셀에 대한 속성을 넣어줄 때 주의점은 원래의 셀 속성 중 colNo 는 넣어주실 필요가 없다는 점입니다. colNo 는 레이아웃상에서 정의 될 때 정해지기 때문에 내부적으로 자동 계산됩니다.

배열에 행 또는 셀의 속성을 지정하려면 SpanArrayCollection 컴포넌트를 사용하며, SpanArrayCollection 의 cellAttributeFieldSuffix 속성을 통하여 Cell 속성이 저장된 dataField 를 찾아 개별 Cell 의 속성을 설정하게 됩니다. 또한 Row 에 대한 속성은 rowAttributeField 속성을 통해 저장된 dataField 를 찾게 됩니다.

```
var dpArray = [
        {Region:"Southwest3",    Region_attr:{rowSpan:2,    colSpan:3,    styleName:"gridSummaryStyle",
backgroundColor:"#00FF00"}, Territory:"Central California",
                Territory_Rep:"Joe Smith",
                Revenues:{Actual:29134, Estimate:30000, Real:30000, Price:3003}},
        {Region:"Southwest", Territory:"Nevada",
                Territory_Rep:"Bethany Pittman",
                Revenues:{Actual:52888, Estimate:45000, Real:30000, Price:3004}},
        {Region:"Nor", Territory:"Arizona",
                Territory_Rep:"Dana Binn",
                Revenues:{Actual:29885, Estimate:30000, Estimate_attr:{rowSpan:2},
            _rattr:{styleName:"allTotalStyle", backgroundColor:"#FFFF00"}},
        {Region:"Nor", Territory:"Central California",
                Territory_Rep:"Joe Smith",
                Revenues:{Actual:29134, Estimate:30000, Real:30000, Price:3001}},
        {Region:"Northwest", Territory:"Southern California",
                Territory_Rep:"Alice Treu",
                Revenues:{Actual:44985, Estimate:45000, Real:30000, Price:3009}}
];
```

<예제 23 배열에 행, 셀 속성 넣은 예>

```
<rMateGrid>
        <DataGrid id="dg1" editable="true" doubleClickEnabled="true"
verticalAlign="middle">
                <columns>
                        <DataGridColumn dataField="Region" width="80"/>
                        <DataGridColumn dataField="Territory" width="150"/>
                        <DataGridColumn dataField="Estimate"/>
                        <DataGridColumn dataField="Real"/>
                        <DataGridColumn dataField="Price" textAlign="right"/>
                </columns>
                <dataProvider>
                        <!--
                         그리드의 자료를 일반 data형식(배열, XML)이 아닌 별도의 컴포넌트에
```

```
입력해야 할 경우 아래와 같이 $gridData를 넣어줍니다
                    -->
                    <SpanArrayCollection source="{$gridData}"/>
            </dataProvider>
        </DataGrid>
</rMateGrid>
```

<예제 24 SpanArrayCollection 컴포넌트 사용 예>

예에서 Region 필드에 대한 Cell 속성 필드명은 Region_attr 이 되며 "_attr"은 SpanArrayCollection 의 cellAttributeFieldSuffix 속성을 통해 변경할 수 있으며, 기본값은 "_attr"입니다. 또한 Row 속성 필드명은 "_rattr"이 되어 해당 Row 에 대한 속성을 정의 하고 있습니다.

데이터에 행, 셀속성을 넣어 처리할 때 콜렉션 컴포넌트에서는 데이터에서 속성을 추출하는 작업이 일어납니다. 그래서 약간의 지연시간이 발생할 수 있으니 유의하시기 바랍니다.

9. Excel Import / Export

rMate Grid 의 Excel Import 와 Export 기능에 대한 설명입니다.
Excel 기능 사용시 일부 브라우저에서 파일을 읽거나 생성할 수 없기 때문에 Import, Export 기능 모두 서버에서 전달된 내용으로 파일을 생성하거나 저장하고 관리하는 모듈이 필요할 수 있습니다. 이때에는 샘플의 ServerSampels 디렉토리에 포함된 예제에 맞춰 upload, download 모듈을 작성하셔야 upload 와 save 시 에 문제없이 기능을 사용하실 수 있습니다. 또한 필요에 따라 서버에서 파일관련된 메모리 설정을 해주셔야 Excel upload, download 에 문제가 발생하지 않습니다. 각각의 서버에 관련된 내용은 아래를 참고 하시기 바랍니다.

9.1. Excel Import (CSV)

Excel Import 에서 Excel CSV 파일 가져오기를 수행할 때에 대한 설명입니다.
CSV Import 기능을 사용할 시 브라우저에 따라 서버에서 CSV 파일을 처리하는 서버 프로그램이 필요합니다. 이때에는 Samples/ServerSamples 폴더에 importCSV.jsp 파일을 참고하시면 JSP 에서 CSV 파일 처리 방식을 확인하실 수 있으며, 서버에서 파일 업로드를 처리할 수 있어야 합니다. (PHP 일 경우에는 importCSV.php 참조)
import 실행시 option 에 따라 화면의 표시내역, 기본값, 레이아웃 변경여부등을 정하게 됩니다. 사용자의 선택이 정상적으로 이루어져 수행이 완료된 후에 importComplete 이벤트가 발생합니다.
Excel CSV Import 사용 시 Parameter 는 아래와 같습니다.

Parameters

- option : 가져오기 옵션을 설정한 Object 이며 설정 가능한 필드는 다음과 같습니다.
  layoutChangeOption:**Number** 레이아웃 변경 방식 (default:0)
    0 : 사용자에게 질의
    1 : 현재 레이아웃에 데이터만 import
    2 : 헤더나 데이터에 따라 레이아웃을 재설정하고 데이터를 import
  headerRowCount:**Number** 헤더라인 수 기본 값 (default:0)
  headerRowCountVisible:**Boolean** 헤더라인 수 표시 여부 (default:true)
  UTF8:**Boolean** UTF8 체크박스에 설정될 기본 값 (default:false)
  UTF8Visible:**Boolean** UTF8 체크박스 표시 여부 (default:true)
  useGroupedColumn : 그룹컬럼 생성 여부. false 일 경우 1 줄의 컬럼만 생성됩니다. (default:true)
  csvDefaultCharSet : csv 파일의 기본 인코딩. (default:"euc-kr")
  parseFunction: **Function** 파싱 처리한 레코드에 대한 검사를 수행하여 레코드를 수정할 수 있는 함수를 지정할 수 있으며 함수의 파라메터는 다음과 같습니다.
    function parseFunction(parsedOb:Objectj, csvStr:String, rowIndex:int):Object
      parsedObj : 파싱된 레코드 object (필드는 F0,F1,F2... 형식으로 저장되어 있음)
      csvStr: 파싱에 사용된 문자열
      rowIndex : 행 번호

- url : 서버에서 CSV 를 Grid 데이터로 가져오도록 하는 프로그램이 위치한 주소 값 입니다. (브라우저가 파일 읽기를 지원하지 않을 경우 사용됨)

9.2. Excel Import (xls, xlsx)

Excel Import 에서 Excel 파일(xls, xlsx)을 읽어들이기 위해서는 오픈 소스인 SheetJS 를 사용하도록 합니다. (따라서 import 에서 지원되는 기능은 SheetJS 에서 지원되는 기능에 한하며, 브라우져별로 지원되는 기능 및 제한(파일 크기 및 행수, 최대 사용가능 메모리등등)이 생기게 됩니다. 주로 export 된 파일을 편집 후 import 하는데 사용하시기 바랍니다)
SheeJS 의 파일은 제품의 rMateGridH5/JS 디렉토리에 있으며 이를 위해 html 에는 다음과 같은 js 가 포함되어야 합니다.

```
<script type="text/javascript" src="../rMateGridH5/JS/jszip.min.js"></script>
<script type="text/javascript" src="../rMateGridH5/JS/xlsx.min.js"></script>
```

위 구문에서 js 파일들의 path 는 설치 경로에 따라 변경되어야 합니다.

Excel 파일을 읽어 들이기 위해 파일 읽기 기능을 사용할 시 예전 브라우저에서는 파일 읽기 기능이 지원되지 않기 때문에 서버에 파일 업로드 후 내려받아 처리하는 방식을 사용해야 하며 이를 처리하기 위

한 서버 프로그램이 필요하게 됩니다. 이때에는 Samples/ServerSamples 폴더의 importExcel.jsp 파일을 참고하시면 JSP 에서 파일 처리방식을 확인하실 수 있습니다. 업로드된 파일의 파일내용을 읽어 들여 base64 로 인코딩하고 파일명과 함께 textarea 에 query string 형식으로 넣어 보내주게 되며, 브라우저에서 base64 문자열을 받아 원래의 파일 내용으로 복원하여 처리하게 됩니다.

import 실행시 option 에 따라 화면의 표시내역, 기본값, 레이아웃 변경여부등을 정하게 됩니다. 사용자의 선택이 정상적으로 이루어져 수행이 완료된 후에 importComplete 이벤트가 발생합니다.
Excel File Import 사용 시 Parameter 는 아래와 같습니다.

Parameters

- option : 가져오기 옵션을 설정한 Object 이며 설정 가능한 필드는 다음과 같습니다.
  layoutChangeOption:**Number** 레이아웃 변경 방식 (default:0)
    0 : 사용자에게 질의
    1 : 현재 레이아웃에 데이터만 import
    2 : 헤더나 데이터에 따라 레이아웃을 재설정하고 데이터를 import
  headerRowCount:**Number** 헤더라인 수 기본 값 (default:0)
  headerRowCountVisible:**Boolean** 헤더라인 수 표시 여부 (default:true)
  useGroupedColumn : 그룹컬럼 생성 여부. false 일 경우 1 줄의 컬럼만 생성됩니다. (default:true)
  parseFunction: **Function** 파싱 처리한 레코드에 대한 검사를 수행하여 레코드를 수정할 수 있는 함수를 지정할 수 있으며 함수의 파라메터는 다음과 같습니다.
    function parseFunction(parsedOb:Objectj, rowIndex:int):Object
      parsedObj : 파싱된 레코드 object (필드는 F0,F1,F2... 형식으로 저장되어 있음)
      rowIndex : 행 번호

- url : 서버에서 엑셀 파일을 Grid 데이터로 가져오도록 하는 프로그램이 위치한 주소 값 입니다. (브라우저가 파일 읽기를 지원하지 않을 경우 사용됨)

9.3. Excel Export (Upload / Save)

Excel Export 는 그리드 데이터를 Excel 형식으로 PC 에 저장하는 기능입니다.
저장할 Excel 파일의 내용은 rMate Grid 내에서 자바스크립트로 만들어지며 브라우저가 HTML5 파일저장 기능을 지원하면 바로 저장되나, 지원하지 않는 브라우저의 경우에는 파일을 생성할 수 없기 때문에 서버를 통해야 첨부파일을 다운로드하는 방식으로 파일을 저장하게 됩니다. 이 때에는 rMate Grid 에서 Excel Export 시 만들어진 파일내용을 base64 형태로 서버에 업로드한 뒤 Excel 로 다운 받는 방식을 사용합니다. Samples/ServerSamples 폴더에 saveExcel.jsp 과 uploadExcel.jsp 파일을 참고하시면 JSP 를 활용한 Excel Export 기능을 확인하실 수 있습니다. 업로드된 base64 형식의 문자열을 decoding 하여 원래의 파일 내용으로 복원한 후 다운로드 시켜주거나 저장하는 처리 방식입니다.

엑셀의 xlsx 형식으로 파일을 처리할 경우에는 오픈 소스인 JSZip 을 사용하도록 합니다. (따라서 지원되는 기능은 JSZip 에서 지원되는 기능에 한하며, 브라우저별로 지원되는 기능 및 제한(파일 크기 및 행수, 최대 사용가능 메모리등등)이 생기게 됩니다)
JSZip 파일은 제품의 rMateGridH5/JS 디렉토리에 있으며 이를 위해 html 에는 다음과 같은 js 가 포함되어야 합니다.

```
<script type="text/javascript" src="../rMateGridH5/JS/jszip.min.js"></script>
```

또한 csv 형식으로 파일을 처리할 경우에는 오픈소스인 js-codepage 를 사용하도록 합니다. js-codepage 파일은 제품의 rMateGridH5/JS 디렉토리에 있으며 이를 위해 html 에는 다음과 같은 js 가 포함되어야 합니다.

```
<script type="text/javascript" src="../rMateGridH5/JS/cpexcel.js"></script>
```

위 구문들에서 js 파일들의 path 는 설치 경로에 따라 변경되어야 합니다.
Export 할 파일의 형식을 dataGrid 의 exportType 속성으로 지정할 경우에는 그에 맞추어 exportFileName 속성의 파일 확장자도 변경되도록 설정해 줘야 합니다.

생성되는 Excel 파일은 특별히 컬럼을 지정하지 않을 경우 현재 화면에 보여지는 컬럼과 동일하게 처리되며 숨겨진 컬럼(visible 속성이 false)이 있을 경우에는 Excel 파일에서도 숨기기 처리됩니다.

ExcelExport 기능은 excelExportSave 와 excelExportUpload 가 있으며 Parameter 는 아래와 같습니다.

Parameters

- url : Grid 데이터를 업로드하고 다운로드할 서버의 모듈이 위치한 주소값 입니다. 기본값 null
- async : 비동기 모드로 수행여부, 기본값 false

async parameter 는 비동기 모드 설정 값입니다. 비동기 모드는 export 데이터가 많을 경우 발생할 수 있는 브라우저 자바스크립트 timeout(수행시간 초과)을 방지하는 기능입니다.

여러 그리드를 하나의 Excel 파일로 export 하려면 excelMultiExportSave 와 excelMultiExportUpload 를 사용할 수 있으며 Parameter 는 아래와 같습니다.

Parameters

- grids : export 를 실행하는 그리드외에 함께 export 할 DataGrid 의 배열을 넣어 줍니다.
- url : Grid 데이터를 업로드하고 다운로드할 서버의 모듈이 위치한 주소값 입니다. 기본값 null
- async : 비동기 모드로 수행여부, 기본값 false

9.4. Excel Import / Export 시 보안 문제

Excel Import 나 Export 시 브라우저의 미지원으로 서버로 데이터를 보내거나 받을 경우 해당 접속 url 은 반드시 html 이 publish 된 서버와 같은 서버여야 정상 작동합니다. 다른 서버(url 의 hostname)를 사용할 경우 브라우저 보안정책에 위배되어 접속이나 접근이 안돼 오류가 발생할 수 있습니다.

10. 다국어 지원

rMate Grid 에서는 오류나 clipboard copy, paste, excel import 등등의 기능 사용시 메시지를 표현하게 되며, 이때 사용될 메시지를 언어에 따라 다른 문장으로 표현할 수 있습니다.

10.1. 그리드 메시지 언어 설정

그리드에서 표시되는 메시지의 언어를 설정하려면 jsVars 에서 (6.1 jsVars 설정 참조) localeChain 을 설정을 하시면 됩니다.
또한 필요에 따라 메시지의 내용을 수정하려면 GridRoot 의 setResourceBundleString 함수를 사용하여 메시지의 내용을 변경하실 수 있으며 내부에서 사용되는 메시지는 다음 장을 참조하시기 바랍니다.
현재 지원되는 메시지의 언어는 한국어(ko-KR), 영어(en-US), 프랑스어(fr-FR), 독일어(de-DE), 중국어 간체(zh-CN) 입니다.

10.2. 그리드 내부 메시지

번들명	메시지명	메시지 내용
rmategridweb	invalidJSON	데이터의 JSON 형식이 잘못되었습니다. 데이터를 확인해 주십시오. 에러:
	invalidXML	데이터의 XML 형식이 잘못되었습니다. 데이터를 확인해 주십시오. 에러:
	invalidCSV	데이터의 CSV 형식이 잘못되었습니다. 데이터를 확인해 주십시오. 에러:
	invalidTSV	데이터의 TSV 형식이 잘못되었습니다. 데이터를 확인해 주십시오. 에러:
	invalidLayoutXML	레이아웃 XML 형식이 잘못되었습니다. 레이아웃의 XML을 확인해 주십시오.
	unknownObject	알 수 없는 개체:{0}을(를) 정의하였습니다.
	unknownVariable	알 수 없는 변수:{0}을(를) 정의하였습니다.
	unknownProperty	알 수 없는 속성:{0}을(를) 정의하였습니다.

	errorSetProperty	속성:{0} 설정 중 오류가 발생하였습니다. 에러내용:
	errorSetBoolean	속성 {0}의 값은 true나 false여야 합니다.
	notExportObject	현재 생성된 DataGrid는 해당 기능을 지원하지 않습니다.
	makeExcelError	엑셀 파일 작성 중 오류가 발생하였습니다. 오류:{0}
	saveExcelError	엑셀 파일 저장 중 오류가 발생하였습니다. 다른 프로그램에서 해당 파일을 사용 중인지 확인해 보시기 바랍니다. 오류:{0}
	uploadExcelError	엑셀 파일 업로드 중 오류가 발생하였습니다. 오류:{0}
	alertTitleExport	Export
	errorUpdateData	데이터 수정에서 오류가 발생하였습니다.
	errorDeleteData	데이터 삭제에서 오류가 발생하였습니다.
	errorConvertData	데이터 변환 중 오류가 발생하였습니다.
	selectDeleteRow	삭제할 행을 선택하세요
	enterSearchString	찾을 문자열을 입력하세요
	needInput	입력이 필요합니다.
	mustEnterInterger	정수를 입력하시기 바랍니다.
	mustEnterNumber	숫자를 입력하시기 바랍니다.
	maxChars	최대 입력 가능 문자수는 {0}자 입니다.
	pasteMustEnterInterger	{0} 컬럼에는 정수를 입력하시기 바랍니다. (값:{1})
	pasteMustEnterNumber	{0} 컬럼에는 숫자를 입력하시기 바랍니다. (값:{1})
	pasteMaxChars	{0} 컬럼의 최대 입력 가능 문자수는 {1}자 입니다. (값:{2})
	errorSortModeAddItem	정렬상태에서는 행을 추가할 수 없습니다.
	errorFilterModeAddItem	필터링상태에서는 행을 추가할 수 없습니다.
datagrid	groupingPanelDrag	컬럼헤더를 이 곳에 끌어다 넣으면 그룹핑됩니다
	importTitle	Load CSV File
	importExcelTitle	Load Excel File
	msgKeepColumns	현재의 컬럼 정보를 그대로 유지하시겠습니까?
	msgNoDataToRead	읽어 들일 정보가 없습니다.
	importAlertTitle	Import
	rowCountLabelText	헤더 라인 수
	sheetSelectLabelText	Import할 Sheet를 선택하세요.
	sheetSelectComboBoxOption	선택
	invalidFileName	잘못된 파일명입니다.
	invalidFileExtension	{0} 형식의 파일만 가능합니다.
	UTF8	UTF-8
shared	dateFormat	YYYY/MM/DD
	monthSymbol	월

	yearSymbol	년
controls	firstDayOfWeek	0
	dayNamesShortest	일,월,화,수,목,금,토
formatters	monthNamesShort	1,2,3,4,5,6,7,8,9,10,11,12
utils	overClipboardLimit	복사하려는 내용이 너무 많습니다. 범위를 줄여주시기 바랍니다. (최대 64000자)
	clipboardCopy	복사하시려면 Ctrl+C를 누르고 Enter키를 누르세요
	clipboardPaste	붙여넣기하시려면 Ctrl+V를 누르세요

11. PivotDataGrid

많은 양의 데이터로 작업 할 때 데이터의 범위와 크기 때문에 애를 먹을 수 있습니다. 예를 들어, 일반적인 2 차원 스프레드 시트에서 서로 다른 제품, 지역 및 고객에 대한 판매 정보를 수집합니다. 스프레드 시트에는 수백 개의 행과 수십 또는 수백 개의 열이 포함될 수 있습니다. 이러한 대규모 데이터 수집에 유용한 정보를 추출하는 것은 어려울 수 있으며 데이터의 추세 또는 다른 패턴을 식별하려고하면 더 어려워 질 수 있습니다.

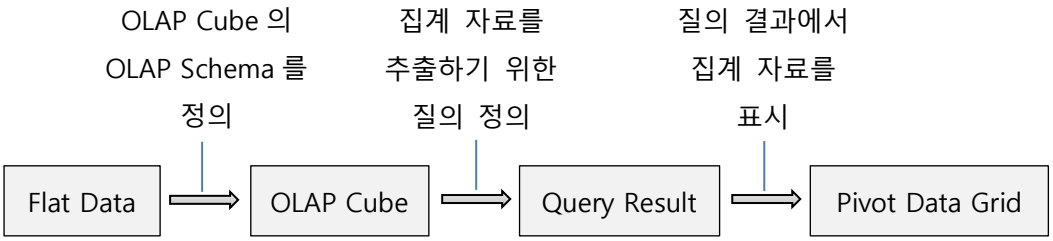
이때 데이터를 분석하기 위해 사용할 수 있는 기능으로 OLAP (online analytical processing) 데이터 그리드와 같이 압축 된 형식으로 데이터를 집계하는 것입니다. PivotDataGrid 는 Microsoft Excel 의 피벗 테이블과 유사합니다. PivotDataGrid 는 데이터 집계를 스프레드 시트와 같이 행과 열의 2 차원 그리드에 표시하지만 집계 설정에 따라 데이터가 압축됩니다.

PivotDataGrid 컨트롤을 사용하여 OLAP 데이터 그리드를 표시합니다. 다음 그림은 product 및 quarter 에 대한 집계 된 판매 정보를 표시하는 PivotDataGrid 컨트롤을 보여줍니다.

Product	Quarter			
	Q1	Q2	Q3	Q4
Apple	420	1430	310	730
Banana	650	1190	1090	2250
Mango	1485	300	785	430
Orange	980	350	1990	590
Pineapple	345	485	715	430

11.1. 피벗 데이터 생성

다음은 PivoDataGrid 에서 데이터를 처리하는 순서를 나타냅니다.



위 과정을 자세히 설명하면

1. 같은 필드를 가지는 행들로 이루어진 플랫 데이터(Flat Data)를 준비합니다.
   아래 코드는 플랫 데이터의 형식에 대한 예입니다.

```
var data = {
      customer:"AAA",
      product:"Apple",
      quarter:"Q1"
      revenue: "100.00"
}
```

2. 일반 자료에서 집계 자료를 어떻게 변환할지를 정의한 OLAP Schema 를 OLAP Cube 에 정의합니다.
   정의되는 OLAP Schema 는 OLAP Cube 에서 플랫 데이터의 표현을 정의하고 OLAP 쿼리에 대한 데이터를 집계하는 방법을 정의합니다.
   OLAP Cube 는 관계형 데이터베이스의 테이블과 유사합니다. 그러나 테이블에는 일반적으로 행과 열의 두 가지 차원이 있지만 OLAP 큐브는 여러 차원을 가질 수 있습니다. 이 예제에서 큐브는 customer, product 및 quarter 의 세 가지 차원으로 구성됩니다. customer, product 및 quarter 에 대한 가능한 모든 값의 집합은 큐브의 고유 한 지점을 정의합니다. 큐브의 각 지점 값은 해당 customer, product 및 quarter 의 집합 값에 대한 revenue 가 됩니다.
3. PivotDataGrid 컨트롤에 표시하기 위해 OLAP Cube 에서 집계 된 데이터를 추출하는 쿼리를 만듭니다.
   데이터가 OLAP Cube 에 있으면 PivotDataGrid 컨트롤에 표시 할 집계 된 데이터를 추출하는 쿼리를 작성합니다. 서로 다른 유형의 데이터 집계를 만들기 위해 여러 개의 쿼리를 작성할 수 있습니다.
4. 쿼리 결과를 PivotDataGrid 컨트롤의 입력으로 사용하여 결과를 표시합니다.

11.2. OLAP Cube

OLAP Cube 는 여러 차원을 가질 수 있습니다. 가장 간단한 형식에서 OLAP Cube 의 *차원(dimension)*은 플랫 데이터 집합의 필드에 해당합니다. 예를 들어 세 가지 필드가 포함 된 플랫 데이터가 있습니다.

```
var data = {
      product:"Apple"
      quarter:"Q1"
      revenue: "100.00"
}
```

각 레코드의 데이터 필드는 다음 값을 포함 할 수 있습니다.

- product 필드의 값은 Apple, Banan, Mango, Orange 및 Pineapple 일 수 있습니다.
- quater 필드는 Q1, Q2, Q3 및 Q4 값을 가질 수 있습니다.
- revenue 필드에는 분기에 대한 제품 판매액이 표시됩니다.

데이터를 집계하려면 quarter 와 product 의 두 가지 차원으로 OLAP Cube 를 만듭니다. 큐브의 각 차원에 따른 값을 *구성원(member)*라고합니다. 예를 들어, 큐브의 product 차원에는 Apple, Banana, Mango 및 Pineapple member 가 있습니다. quater 차원의 member 는 Q1, Q2, Q3 및 Q4 입니다.

두 차원으로 정의 된 큐브의 모든 지점에서 값을 큐브의 *측정값(measure)*이라고 합니다. 예를 들어, (Q1, Apple)에 의해 정의 된 큐브의 지점에서 측정값은 100.00 입니다. Schema 는 OLAP Cube 의 단일 지점에 대해 하나 이상의 측정값을 정의 할 수 있습니다.

OLAPCube 는 수치에 대해서만 measure 를 지원합니다. 숫자 값의 장점은 PivotDataGrid 컨트롤에 표시하기 위해 쉽게 집계 할 수 있다는 것입니다. 일반적인 집계 유형에는 합계, 평균, 최소 및 최대가 있습니다. 예를 들어 revenue 측정값의 집계 방법을 SUM 으로 지정합니다. 그런 다음 Apple 큐브에서 판매 정보를 추출합니다. 집계 된 판매 데이터에는 각 분기에 대한 모든 Apple 판매액의 합계가 포함됩니다.

11.3. OLAP Schema

플랫 데이터를 OLAP Cube 로 변환하려면, 플랫 데이터의 필드를 지정하여 차원의 구성원을 제공하는 큐브의 차원(dimension)과 플랫 데이터의 필드로 큐브내의 임의의 점에 대한 값을 제공하는 측정값(measure)으로 OLAP Schema 를 만듭니다.
예를 들어 판매 정보가 포함 된 다음과 같은 간단한 데이터가 있습니다.

```
var data = {
      customer:"AAA",
      product:"Apple",
      quarter:"Q1"
      revenue: "100.00"
}
```

다음 예에서는 큐브에서 이 데이터를 나타내는 데 사용되는 OLAP Schema 의 정의가 포함 된 OLAPCube 에 대한 정의를 보여줍니다. 이 스키마는 customer, product 및 quarter 필드를 기반으로 3 차원 OLAP Cube 를 정의합니다.

```
<OLAPCube name="FlatSchemaCube"
        dataProvider="{flatData}"
        id="myCube">
        <OLAPDimension name="CustomerDim">
                <OLAPAttribute name="Customer" dataField="customer"/>
                <OLAPHierarchy name="CustomerHier" hasAll="true">
                        <OLAPLevel attributeName="Customer"/>
                </OLAPHierarchy>
        </OLAPDimension>
        <OLAPDimension name="ProductDim">
                <OLAPAttribute name="Product" dataField="product"/>
                <OLAPHierarchy name="ProductHier" hasAll="true">
                        <OLAPLevel attributeName="Product"/>
                </OLAPHierarchy>
        </OLAPDimension>
        <OLAPDimension name="QuarterDim">
                <OLAPAttribute name="Quarter" dataField="quarter"/>
                <OLAPHierarchy name="QuarterHier" hasAll="true">
                        <OLAPLevel attributeName="Quarter"/>
                </OLAPHierarchy>
        </OLAPDimension>
        <OLAPMeasure name="Revenue"
                dataField="revenue"
                aggregator="SUM"/>
</OLAPCube>
```

- 모든 차원(dimension)이 먼저 정의되고 모든 측정값(measure)이 정의됩니다.
- 각 차원의 첫 번째 행은 플랫 데이터의 데이터 필드를 OLAPAttribute 인스턴스와 연결합니다. 그런 다음 OLAPLevel.attributeName 특성을 사용하여 특성을 차원과 연관시켜 차원 구성원을 채

웁니다. 예를 들어, 이 스키마에서는 CustomerDim 차원의 Customer level 을 데이터의 customer 필드의 데이터로 채웁니다.

- OLAP Schema 의 차원에는 항상 하나 이상의 계층 구조가 포함됩니다. 이 스키마에서 각 차원의 계층 구조에는 플랫 데이터의 필드에 해당하는 단일 수준의 level 만 포함됩니다. 이것은 차원의 가장 단순한 형식입니다. 다른 스키마는 계층 구조에서 여러 level 을 정의하여 복잡한 차원을 만들 수 있습니다.
- 측정값 정의는 OLAP 큐브의 각 지점 값과 쿼리에서 측정값을 집계하는 방법을 포함하는 플랫 데이터의 데이터 필드를 지정합니다. 이 예에서는 revnue 를 합산하여 집계합니다. 즉, 이 OLAP Cube 의 모든 쿼리는 revenue 합계를 반환합니다. 다른 유형의 집계 메소드에는 최대값, 최소값, 평균이 포함됩니다.
- OLAPCube 에는 complete 이벤트에 대한 이벤트 핸들러를 지정할 수 있습니다. 큐브에서 초기화가 완료 할 때까지는 쿼리를 호출 할 수 없으며, complete 이벤트를 발생후에만 쿼리를 수행해야 합니다.

응용 프로그램의 요구 사항에 따라 동일한 플랫 데이터 세트에서 여러 OLAP Cube 를 만들 수 있습니다. 각 OLAP Cube 는 서로 다른 스키마를 사용하여 고유 한 차원과 차원의 배열을 만듭니다.

12. KeyBoard 입력

rMate Grid 는 키보드를 통하여 값을 입력하거나 커서를 이동할 수 있습니다.

12.1. KeyBoard

rMate Grid 에서 사용하실 수 있는 키보드 입력은 다음과 같습니다.

- **방향키** : 셀과 행에서 상하좌우로 이동할 수 있습니다.
- **Page Up/Down** : 그리드의 페이지 업 다운으로 행을 페이지 만큼 이동할 수 있습니다.
- **Shift Page Up/Down** : selectionMode 가 singleRow, multipleRows 일 경우 그리드의 컬럼을 좌우로 스크롤할 수 있습니다.
- **Home/End** : 그리드의 첫행과 마지막행으로 이동합니다.
- **ESC** : Data Edit 시에 editing 을 취소하고 변경 전으로 돌아가며 포커스가 취소됩니다.
- **Enter** : Data Edit 시에 수정된 데이터를 입력하며 데이터가 올바르게 입력된 경우 아래의 행으로 이동합니다. Shift + Enter 입력시 위의 행으로 이동합니다.
- **Tab** : Data Edit 시에 수정된 데이터를 입력하며 데이터가 올바르게 입력된 경우 우측의 행으로 이동합니다. Shift + Tab 입력시 좌측의 행으로 이동합니다.
- **F2** : DataGrid 의 selectionMode 가 singleCell 이나 multipleCells 이고 Data Edit 가 가능할 경우에만 동작하며 키 입력시 Edit 모드로 변경 됩니다.
- **Space** : 선택의 위치가 헤더일 경우 해당 컬럼이 소팅 가능하면 소팅이 일어나게 됩니다.

- **Shift + 마우스 클릭** : selectionMode 가 multipleCells, multipleRows 일 경우 마우스로 셀 선택시 Shift 를 누르면 선택된 셀에서 다음 클릭한 셀까지 선택 영역으로 지정됩니다.
- **Ctrl + 마우스 클릭** : selectionMode 가 multipleCells, multipleRows 일 경우 마우스로 셀, 행 선택시 Ctrl 을 누르면 각각의 셀과 행이 개별적으로 선택 됩니다.
  헤더를 클릭할 경우에는 다중 정렬기능으로 기존 정렬에 추가로 정렬을 더하는 작업이 됩니다.
- **Shift + Ctrl + 마우스 클릭** : 헤더를 클릭할 경우 해당 컬럼이 소팅된 상태이고 DataGrid 의 sortExpertMode 가 true 일경우 소팅을 해제합니다.
- **Shift + 방향키** : selectionMode 가 multipleCells, multipleRows 일 경우 셀 선택시 Shift 를 누르고 방향키로 이동하면 이동한 셀들이 선택 영역으로 지정됩니다.
- **Ctrl + 방향키** : Ctrl 키를 누르고 방향키로 이동하면 선택없이 행이나 셀간을 이동하게 되며 캐럿만 표시됩니다.
- **Ctrl + Space** : Ctrl + 마우스 클릭으로 선택한 것과 같은 기능이 발생됩니다. DataGrid 의 selectionMode 가 multipleCells, multipleRows 모드 시 Ctrl 을 누른 상태에서 방향키를 움직여서 원하는 셀로 이동한 후 Ctrl + Space 를 누르게 되면 해당 셀, 행이 개별 선택 됩니다.
  헤더가 선택된 상태에서 입력할 경우에는 소팅이 일어나거나 소팅의 순서가 바뀌게 됩니다.

rMate Grid 가 계층형 데이터일 경우 사용하실 수 있는 키보드 입력은 다음과 같습니다.

- **숫자패드 +** : 선택된 노드를 엽니다.
- **숫자패드 -** : 선택된 노드를 닫습니다.
- **숫자패드 *** : 선택된 노드가 열려있으면 닫고 닫혀있으면 엽니다.
- **Shift + Ctrl + 오른쪽 방향키** : 선택된 노드를 엽니다.
- **Shift + Ctrl + 왼쪽 방향키** : 선택된 노드를 닫습니다.

DateEditor(날짜 입력 에디터) 에서 사용할 수 있는 키보드 입력은 다음과 같습니다.

- **방향키** : 표시된 달내에서 일자를 상하좌우로 이동할 수 있습니다.
- **Page Up/Down** : 이전달, 다음달로 이동할 수 있습니다.
- **Home/End** : 1 일과 해당달의 마지막날로 이동합니다.
- **+/-** : 다음년, 이전년으로 이동합니다.
- **Enter** : 선택한 일자로 수정을 마칩니다.
- **Alt + 윗쪽 방향키** : 표시된 달력 팝업을 숨깁니다.
- **Alt + 아래쪽 방향키** : 달력 팝업을 표시합니다.

MonthEditor(년월 입력 에디터) 에서 사용할 수 있는 키보드 입력은 다음과 같습니다.

- **방향키** : 표시된 년내에서 월을 상하좌우로 이동할 수 있습니다.
- **Page Up/Down** : 다음년, 이전년으로 이동할 수 있습니다.

- **Home/End** : 1 월과 12 월로 이동합니다.
- **+/-** : 다음년, 이전년으로 이동합니다.
- **Enter** : 선택한 년월로 수정을 마칩니다.
- **Alt + 윗쪽 방향키** : 표시된 달력 팝업을 숨깁니다.
- **Alt + 아래쪽 방향키** : 달력 팝업을 표시합니다.

13. 컬럼의 크기에 관하여

rMate Grid 에서 컬럼의 크기는 지정하지 않거나, width 속성을 통해 지정하거나 defaultWidth 속성을 통해 지정하는 것이 가능합니다.
이때 width 를 통해 지정하는 것은 컬럼의 크기를 해당 크기로 고정시켜서 되도록 해당 크기를 유지하라는 뜻이 되며, deaultWidth 를 지정하는 것은 컬럼의 폭은 가변일 수 있으나 설정 값을 참조하여 비율을 지켜서 늘리라는 뜻이 되고, 아무 것도 지정하지 않으면 기본은 100 픽셀이 설정되나 필요시 적당한 크기로 변해도 된다는 뜻이 됩니다.

rMate Grid 는 좌우 스크롤시(horizontalScollPolicy 가 auto 또는 on 일 경우) 픽셀단위가 아닌 엑셀과 같이 컬럼 단위로 이동합니다.
이에 따라 마지막 컬럼이 보여지는 상태에서, 그리드의 폭과 보여지는 컬럼들의 폭의 합과 차이가 발생하는 경우 여백을 어떻게 처리할지를 지정할 수 있습니다.
이는 DataGrid 의 lastColumnWidthPolicy 속성을 통해 조정하며 다음 값을 가질 수 있습니다.

- **balance** : 보여지는 칼럼들의 폭의 합과 그리드 폭과 차이나는 부분을, 마지막 칼럼이 표시될 때 함께 표시되는 칼럼 중 크기(width)가 설정되지 않은 칼럼들에 균등하게 배분합니다. 하지만 모든 컬럼에 크기가 설정되어 있으면 맨 마지막 컬럼의 크기를 강제로 늘리게 되어 expand 로 설정한 것과 같은 모양이 됩니다.
- **expand** : 맨 마지막 컬럼의 크기를 늘려서 표시합니다.
- **cut** : 마지막 컬럼의 크기만큼만 표시하고 나머지는 비워두게 됩니다.

lastColumnWidthPolicy 의 기본값은 balance 입니다.

horizontalScollPolicy 가 off 이거나 컬럼들의 폭의 합이 그리드의 폭보다 작으면, 남는 여백에 대해 크기가 설정되지 않았거나 defaultWidth 가 설정된 컬럼을 우선하여 폭을 조정하게 됩니다.
다만 모든 컬럼이 width 가 지정된 경우에는 horizontalScollPolicy 가 off 일 경우에는 모두 무시하고 균등하게 크기를 나누어 설정하게 되며, auto 또는 on 일 경우에는 맨 마지막 컬럼의 크기를 늘려 처리하게 됩니다.

14. rMate Grid 의 초기 설정 변경

rMateGridH5.setConfig 기능을 이용하여 그리드의 초기값을 변경시킬 수 있습니다.

변경할 수 있는 초기값으로는 DataGrid, DataGridColumn 등의 클래스의 스타일, 상수, 속성과 그리드 내의 언어별 메시지(ResourceBundle), rMateH5.css 로 정의된 스타일입니다. 이때 rMateH5.css 의 내용을 변경시키는 것은 되도록이면 rMateH5.css 의 내용을 직접 수정하여 적용하는 것이 효율적이며, rMateGridH5.setConfig 로 변경시키는 것은 css 의 양에 따라 시간이 걸릴 수 있습니다.
rMateGridH5.setConfig 를 통해 초기값을 변경시키려면 html 의 head 부에 rMateGridH5.js 와 rMateH5.css 가 include 된 후에 rMateGridH5.setConfig 를 실행시킬 js 를 포함시켜 html 이 로딩되기 전에 rMateGridH5.setConfig 을 실행시키면 됩니다.
변경할 내용은 자바스크립트의 Object 형식으로 설정하여 파라메터로 넘기면 되며, Object 의 형식은 다음과 같습니다.

```
클래스명 : {                              // 개별 클래스의 기본값 변경
    Styles : {
        스타일명 : 값
    },
    Properties : {
        속성명 : 값
    },
    Constants : {
        상수명 : 값
    },
},
ResourceBundle : {                        // 그리드내의 메시지
    메시지의 언어 locale : {
        번들명 : {
                메시지명 : 메시지
        }
    }
},
Styles : {                                // rMateH5.css 의 스타일
    스타일의 Selector : {
        스타일명 : 값
    }
}
```

다음은 위의 형식에 따라 적용한 예입니다.

```
var rMateGridH5.newConfig = {
    DataGrid : {
```

```
        Styles : {
            alternatingItemColors: ["#FFFFFF", "#F9F9F9"],
            fontSize : "12px",
            border : "#555555 solid 1px",
            paddingBottom : 3,
            textSelectedColor: "#2B333C"
        },
        Properties : {
            horizontalScrollPolicy : "auto",
            draggableColumns : false
        }
    },
    DataGridSortItemRenderer : {
        Constants : {
            UP_ICON_NAME : "up-arrow-white.png",
            DOWN_ICON_NAME : "down-arrow-white.png"
        }
    },
    ResourceBundle : {
        "ko-KR" : {
            shared : {
                "dateFormat":"YY/MM/DD"
            },
            controls : {
                "firstDayOfWeek" : 1,
                "dayNamesShortest" : "월,화,수,목,금,토,일"
            }
        },
        "en-US" : {
            shared : {
                "dateFormat":"YYYY/MM/DD"
            }
        }
    },
    Styles : {
        rMateDataGridHeaderStyles : {
            color : "#FFFFFF"
        }
```

```
    }
};
```

이에 대한 예제는 샘플의 rMateGridH5/JS 디렉토리 아래에 rMateGridH5.style31.js 와 Samples 디렉토리의 SetConfig_Style3.html 를 참조하시기 바랍니다.
그리드 내의 언어별 메시지(ResourceBundle)에 대한 설명은 10.2 그리드 내부 메시지 를 참조하시고, 만일 이외의 메시지를 변경시킬 필요가 있다면 리아모어 소프트로 문의 바랍니다.

15. rMate Grid 의 사용시 유의할 점

rMate Grid 는 모바일 및 로컬에서 활용이 가능합니다. 하지만 기능 지원에 대하여 몇가지 차이점이 있습니다.

15.1. 모바일 기기에서 사용시

그리드를 모바일에서 사용할 경우 모바일 브라우저의 특성상 다음과 같은 기능은 지원이 되지 않습니다. 컬럼이동, 데이타팁(툴팁), context 메뉴(특정 안드로이드 브라우저에서는 가능, iOS 미지원), Excel Export(특정 브라우저에서만 가능), 드래그 기능을 이용한 선택

15.2. 로컬에서 그리드 사용시

그리드를 로컬에서 사용할 경우 브라우저의 보안으로 인하여 Excel 의 기능을 정상적으로 이용하실 수 없습니다.

15.3. JavaScript Timeout 문제

Excel Import 나 Export, data loading(parsing)시 브라우저내에서 자바스크립트로 작업을 할 경우 데이터가 많아 자바스크립트 timeout(수행시간 초과)이 발생하여 계속 실행여부를 묻는 창이 뜰 수 있습니다. 이는 브라우저별로 차이가 있으나 브라우저의 정책이므로 변경이 불가하며 계속 실행을 선택하다면 기능에 문제가 생기지는 않습니다. 따라서 데이터가 많거나 사양이 낮은 컴퓨터의 사용이 예상되어 timeout 이 발생할 소지가 있을 경우에는 사용자에게 "계속 실행을 선택하도록" 별도로 가이드를 하여야 합니다.

15.4. 그리드를 이용한 에디팅

그리드의 데이터가 그룹핑일 경우에는 에디팅은 지원되지 않습니다.
소팅 또는 필터링 상태에서는 입력후 정렬이나 필터링 상태에 따라 행이 이동하거나 사라질 수 있으므로 에디팅시 유의하시기 바랍니다.

15.5. 그리드의 복사, 붙여넣기

그리드에서 지원되는 복사, 붙여넣기 기능은 셀의 값에 대한 것으로 셀의 속성을 복사하거나 엑셀의 속성을 붙여넣기 하는 기능은 지원되지 않습니다.

15.6. 좌우 스크롤시 그리드의 컬럼 크기

그리드에서 수평 스크롤시 컬럼단위로 이동하기 때문에 마지막 컬럼이 보여질 때 여백의 문제가 발생할 수 있습니다. 이에 대해 자세한 사항을 알고 싶으시면 13 컬럼의 크기에 관하여를 참조 바랍니다.

15.7. 스타일의 별도 지정

그리드에서 사용되는 스타일은 rMateH5.css 파일에 정의되어 있으며, 기본값을 조정하여 원하시는 모양으로 변경이 가능합니다.
하지만 내용을 변경하거나 이 파일의 내용을 별도의 파일에 포함시켜 관리하실 경우에는 다음 사항에 유의하여 작성하시기 바랍니다.

- 사용되는 스타일명은 시스템안에서 중복되지 않도록 해야 합니다.
- 별도의 계층을 만들어 사용하면 안됩니다.
  가령 .rMateH5__DataGrid 를 .grid .rMateH5__DataGrid 와 같이 계층안에 포함시킬 경우 오작동의 원인이 될 수 있습니다.
- margin 요소는 사용하지 마십시오.
- em 단위를 사용하지 마십시오. (px 단위를 사용하시기 바랍니다)

15.8. rMateH5.css 파일의 위치

html 에 포함되는 rMateH5.css 는 rMate Grid 프로그램에서 접근하여 조작되는 경우가 있는데, 크롬, 파이어폭스등에서는 다른 서버에서 읽어들인 css 에 대해 조작시 보안 문제가 발생할 수 있으니 rMateH5.css 는 되도록 같은 서버에서 읽어들이는 것을 권장합니다.
또한 html 내의 CSS 중에서 rMateH5.css 를 찾는데 시간이 걸릴 수 있으므로 많은 CSS 가 사용되는 경우 rMateH5.css 를 위쪽에 위치시키는 것이 성능향상에 도움이 됩니다.

15.9. 다운로드 데이터의 크기 제한

setDataURLEx 함수 등을 통해 외부의 데이터를 로딩할 경우 브라우져의 한계(처리가능 문자열의 최대 크기)에 따라 읽어 들여 처리할 수 있는 크기에 제한이 생깁니다. 이는 브라우져 별로 다를 수 있으며 제한 용량은 다음과 같습니다.

브라우져	최대크기	비고
크롬	512 MiB	V8엔진, 32Bit(윈도우의 경우)
파이어폭스	2 GiB	65버젼 이하는 512MiB
사파리	4 GiB	

참고 : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/length

<감사합니다>