RealGrid 개발자 가이드 (Tutorial A Class)

Mar 3, 2015

RealGridJS 주요기능

Mar 4, 2015

A0 RealGrid 평가판 받기

Mar 29, 2015

A1 RealGridJS 설치하기 (v1.0.10 이상)

Jun 17, 2015

A2 컬럼 만들기

Jun 26, 2015

A3 컬럼이름 바꾸기

Jun 26, 2015

A4 DataProvider에 Field 만들기

Jun 29, 2015

A5 컬럼-필드 연결하기

Jun 30, 2015

A6 RealGrid에 데이터 넣기

Jun 30, 2015

A7 필드 하나에 컬럼 두 개 연결하기

Jul 6, 2015

A8 포커스셀(Focused Cell) 이해하기

Jul 6, 2015

A9 여러행 데이터와 RowId 이해하기

Jul 9, 2015

A10 ItemModel과 ItemIndex 이해하기

Jul 9, 2015

A11 Data와 Item의 다른점은?

Jul 13, 2015

A12 소팅(sorting), 데이터 정렬하기 - I 단일 컬럼 정렬

Jul 14, 2015

A13 소팅(sorting), 데이터 정렬하기 - II 다중 컬럼 정렬

Jul 15, 2015

A14 소팅(sorting), 데이터 정렬하기 - III orderBy()함수 사용하기

Jul 15, 2015

A15 로우 그룹핑(row grouping) - I 드래깅(dragging)을 이용하여 그룹핑

Jul 15, 2015

A16 로우 그룹핑(row grouping) - II groupBy()함수로 그룹핑

Jul 27, 2015

A17 행과 열 고정하기(Fixing)

Jul 29, 2015

A18 컬럼 필터링(Filtering) - I 필터선택상자 사용하기

Jul 30, 2015

A19 컬럼 필터링(Filtering) - II setColumnFilters()함수 사용하기

Jul 31, 2015

A20 셀 선택하기(Selecting) - SelectOptions

Aug 5, 2015

A21 셀 선택하기(Selecting) - 선택한 셀의 값 가져오기

Aug 7, 2015

A22 셀 선택하기(Selecting) - 동적으로 셀 선택하기

Aug 10, 2015

A23 데이터 편집하기(Editing)

Aug 11, 2015

A24 데이터 불러오기(Data Loading)

Aug 12, 2015

A25 에디터(Editor)와 셀의 값

Aug 17, 2015

A26 이벤트 이해하기(Events)

Aug 20, 2015

A27 옵션(Options)의 종류와 설정 방법

RealGridJS 주요기능

Mar 3, 2015

RealGridJS RealGrid 기능정의 Features

RealGridJs 주요기능 정리

- 컬럼(Column)
  - 컬럼 너비 조정
  - 컬럼 너비 자동 조정
  - 컬럼 틀 고정
  - 컬럼 숨기기
  - 컬럼 스타일 변경
  - 컬럼 이동
  - 컬럼 헤더 호버링
  - 컬럼 레이아웃 파일 저장
  - 컬럼 그룹핑
  - 다중 정렬
  - 셀 병합
  - 필터링
  - 필터 핸들러
- 로우(Row)
  - 행 높이 조정
  - 행 틀 고정
  - 행 스타일 변경
  - 행 선택
  - 그룹핑
- 편집(Editing)
  - 다양한 편집 기능
  - 데이터 유효성 체크
  - 셀 또는 행 단위 편집 가능 여부 조정
  - 컬럼 포멧 적용
  - 멀티 콤보 필터
- 데이터(Data)
  - 데이터 검색
  - 다중 그리드 뷰 기능
  - 레이지 로딩 지원
  - 선택된 셀들의 집계 연산 기능
  - 데이터 롤백
  - 전체 데이터 선택
- 엑셀(Excel)
  - 엑셀 내보내기

    - 엑셀 들여오기
- 확장기능(Extention)
    - 합계/평균 및 소계 기능
    - 대용량 데이터 처리
    - 언어변환 가능 여부
    - 멀티라인 그리드
    - 복사 및 붙여넣기
    - 그리드 내부에 그래프 표현
    - 바코드 표현
    - 선택 이벤트
    - 스타일 변경 가능

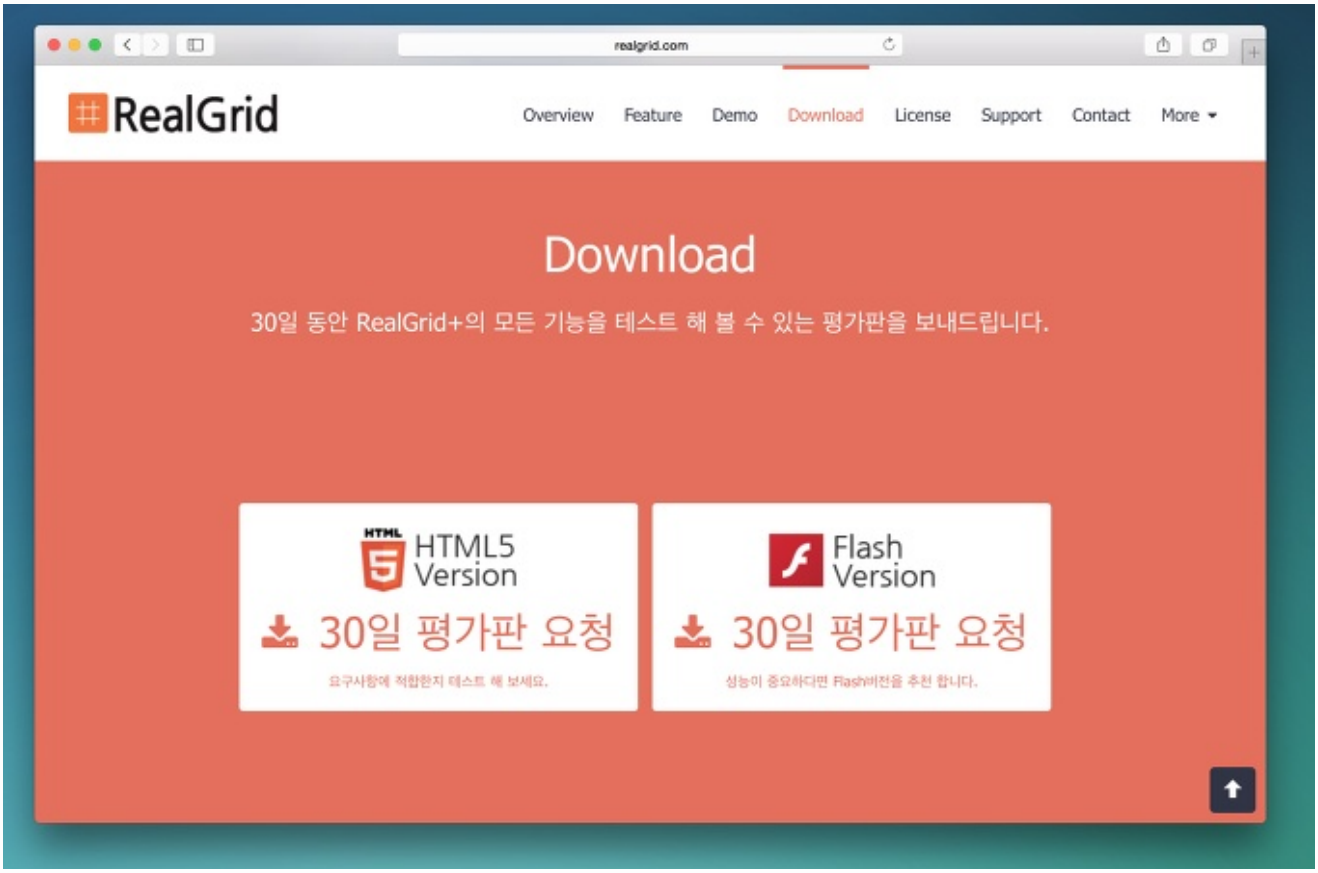
참조

- RealGrid API

A0 RealGrid 평가판 받기

Mar 4, 2015

RealGridJS RealGrid Tutorial 평가판 evaluation



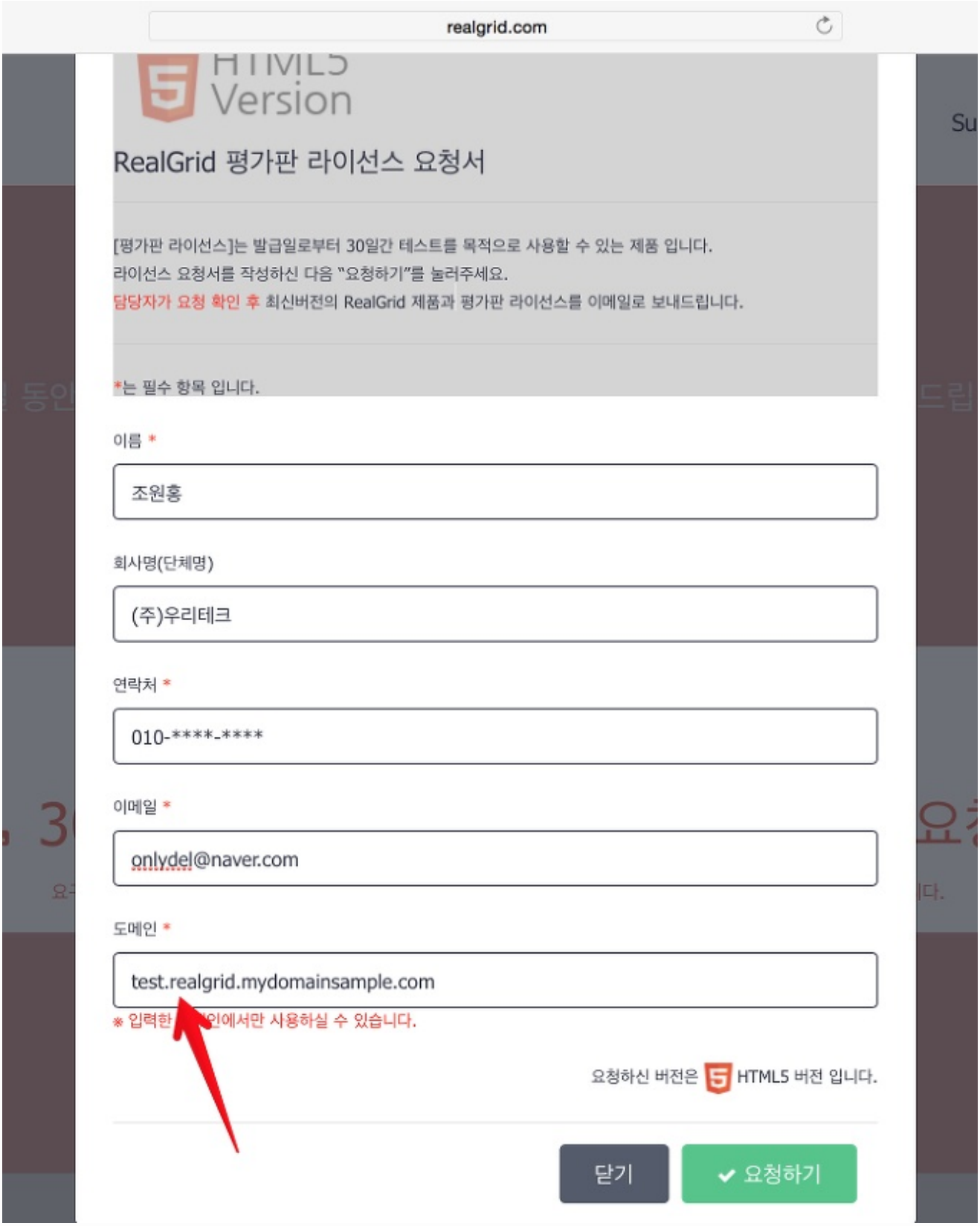
들어가며

이 글은 RealGridJS버전 평가판을 받는 방법에 대해 포스팅한 글입니다.

RealGrid+와 RealGridJS는 둘 다 RealGrid의 홈페이지에서 평가판을 받을 수 있습니다. 하지만, 아직 평가판을 직접 다운로드 할 수 있는 방법은 없습니다. 홈페이지에서 평가판을 요청 하면 이메일을 통해 평가판을 받을 수 있습니다.

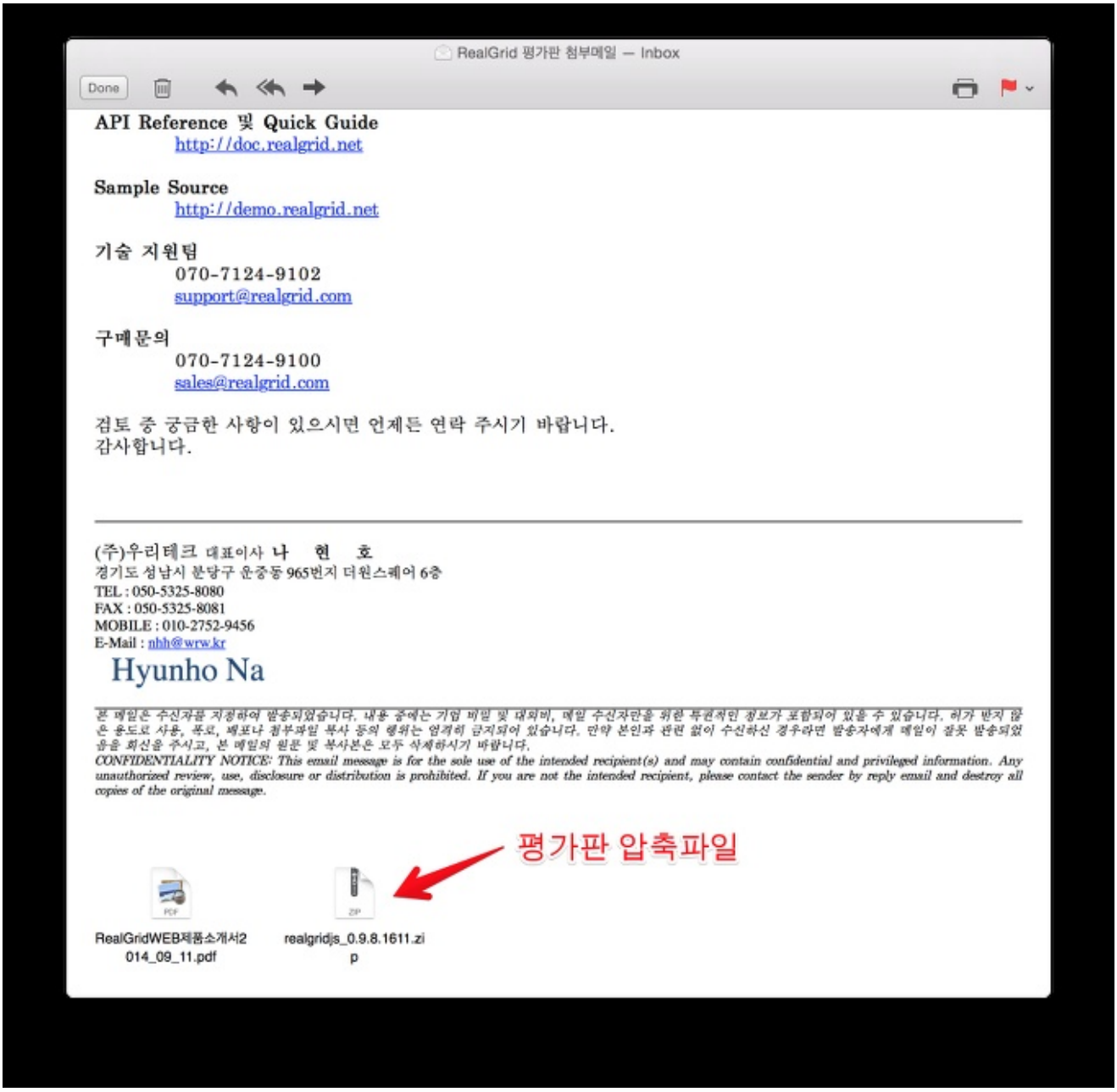
평가판 요청

평가판을 다운로드 하기 위해 홈페이지로 들어가 보겠습니다. 홈페이지에서 Download메뉴를 클릭하면 평가판을 요청 할 수 있는 페이지로 이동합니다. 30일 동안 사용할 수 있는 평가판은 HTML5버전인 RealGridJS와 Flash 버전인 RealGrid+를 따로 요청 할 수 있도록 버튼이 마련되어 있습니다. 평가판 요청 버튼을 눌러보면 평가판요청에 필요한 기본 정보를 입력할 수 있는 화면이 팝업됩니다.

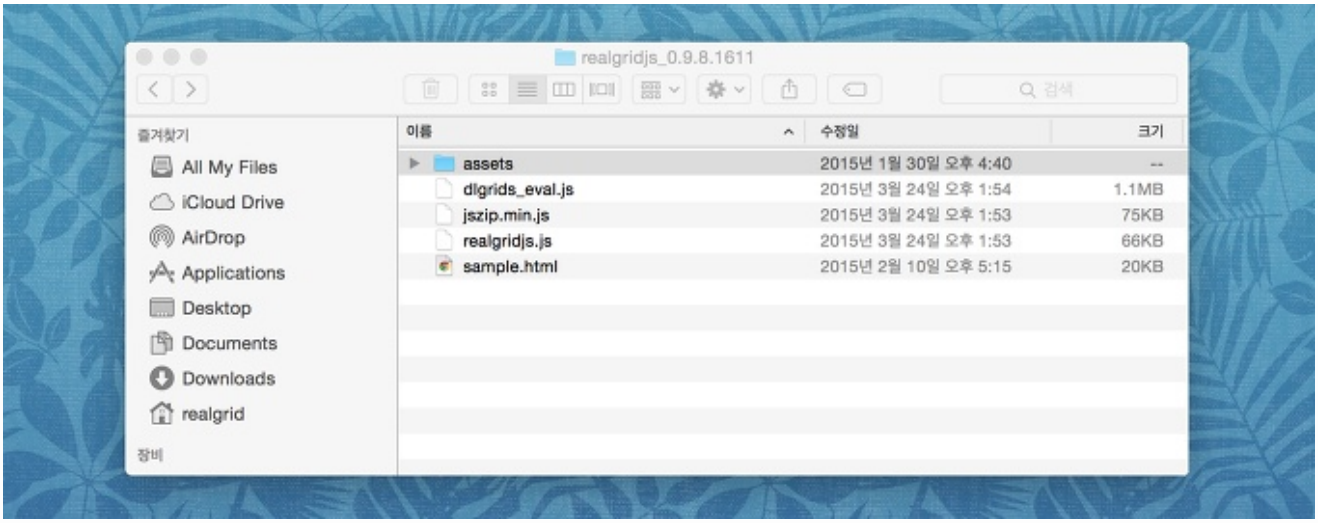


이름, 회사명, 연락처, 이메일을 입력하고 RealGrid를 사용할 도메인을 입력해 주어야 합니다. RealGrid는 도메인 단위로 동작되는 라이선스 방식을 채택하고 있기 때문에 해당 도메인 이외의 도메인에서는 사용할 수 없습니다. 그러니 반드시 평가판을 테스트해 볼 도메인 명을 입력 해야 합니다. "localhost"이나 IP주소를 입력해도 된다고 합니다.

정보를 모두 입력 했다면 요청하기 버튼을 누르면 됩니다. 이제 조금 기다리면 메일을 통해 요청한 평가판이 발송되어 옵니다.



메일로 받은 파일을 열어보면 아래 화면과 같이 하나의 폴더와 네 개의 파일로 구성되어 있습니다. 우선 assets폴더는 RealGrid에서 사용되는 이미지 리소스가 들어 있는 폴더 입니다. 버튼 아이콘 이미지가 들어 있습니다. 네 개의 파일중 sample.html파일은 RealGrid를 웹에서 볼 수 있는 샘플 파일입니다. 나머지 JavaScript파일들은 RealGrid 컨트롤을 웹 화면에 올리는데 필요한 핵심 파일들입니다. javascript폴더에 넣고 사용할 페이지 소스에서 링크를 걸어 사용하면 됩니다.



참조

- RealGrid 평가판 다운로드

A1 RealGridJS 설치하기 (v1.0.10 이상)

Mar 29, 2015

RealGridJS RealGrid Setup

들어가며

이번 강좌에서는 RealGridJS의 설치에 대해 배워보겠습니다. 이 강좌의 내용에 해당하는 버전은 1.0.10 이후 버전입니다.

> 1.0.10 버전 보다 이전 버전의 설치 방법은 [v1.0.10 이전 버전 설치하기 (v1.0.9.1988 이하)]강좌를 참조하세요.

이론

RealGrid를 개발 환경에 맞게 설치해 보겠습니다. 만약, RealGrid가 없다면 평가판요청페이지에서 평가판을 요청하면 메일로 평가판을 받을 수 있습니다. 메일에 포함된 RealGrid파일중 설치에 반드시 필요한 파일은 아래 다섯 개의 Javascript파일들과 RealGridJS화면 구성에 필요한 assets폴더에 들어있는 이미지 파일들 입니다.

Javascript파일중 jszip.min.js파일은 엑셀파일 Import/Export에 필요한 파일이므로 엑셀파일 내보내기 기능이 필요한 화면에서는 반드시 포함시켜주셔야 합니다.

참고로 RealGridJS는 JQuery와 같은 외부 라이브러리가 필요없습니다.

RealGridJS 평가용/개발자용 버전 파일

```
/scripts/assets/
/scripts/realgridjs-lic.js
/scripts/realgridjs_eval.{version}.min.js
/scripts/realgridjs-api.{version}.js.js
/scripts/jszip.min.js
```

RealGridJS 운영용 버전 파일

```
/scripts/assets/
/scripts/realgridjs-lic.js
/scripts/realgridjs.{version}.min.js
/scripts/realgridjs-api.{version}.js.js
/scripts/jszip.min.js
```

실습

이제 RealGridJS를 웹 화면에 올려 보겠습니다.

1. 세 개의 스크립트파일을 순서대로 include합니다. 반드시 아래의 순서대로 파일을 불러와야 합니다.

```
<script type="text/javascript" src="/scripts/realgridjs-lic.js"></script>
<script type="text/javascript" src="/scripts/realgridjs_eval.1.0.14.min.js"></scri
<script type="text/javascript" src="/scripts/realgridjs-api.1.0.14.js"></script>
```

2. GridView객체를 저장할 gridView변수를 정의 합니다.

```
var gridView;
```

3. LocalDataProvider객체를 저장할 dataProvider변수를 정의 합니다.

```
var dataProvider;
```

4. assets폴더의 위치를 변경하기 위해 setRootContext(path)함수를 호출 합니다. 이때 `assets` 이라는 폴더 이름은 변경할 수 없습니다. 반드시 path에 지정된 경로 아래에 `assets` 이라는 이름의 폴더가 존재해야 하며, 그 아래에 화면을 구성하기 위한 이미지 파일들이 있어야 합니다.

```
RealGridJS.setRootContext("/script");
```

5. LocalDataProvider와 GridView객체를 생성하고 GridView의 DataSource를 생성된 LocalDataProvider로 지정하는 코드를 추가 합니다.

```
dataProvider = new RealGridJS.LocalDataProvider();
gridView = new RealGridJS.GridView("realgrid");
gridView.setDataSource(dataProvider);
```

6. RealGridJS가 표시될 `div` 태그를 작성하고 크기를 지정해야 합니다. 크기가 지정되지 않으면 화면에 RealGridJS가 표시되지 않습니다.

```
<div id="realgrid" style="width: 100%; height: 200px;"></div>
```

A2 컬럼 만들기

Jun 17, 2015

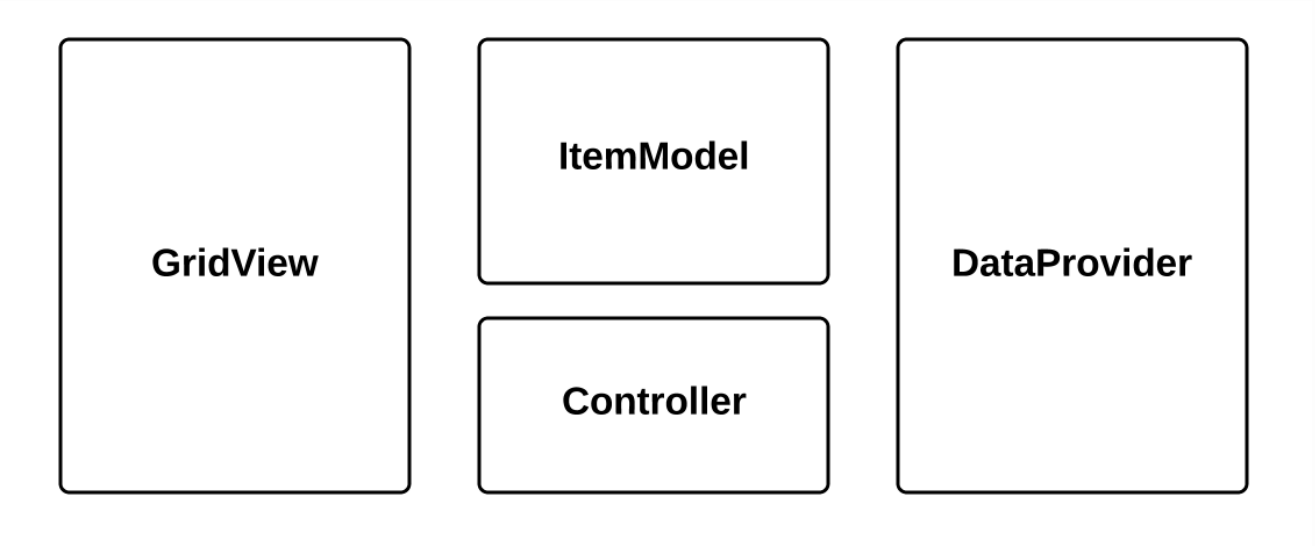
RealGridJS RealGrid column

들어가며

[A1 강좌]에서 html을 이용해 화면에 RealGridJS를 올리는 방법에 대해 알아 보았습니다. 이 강좌에서는 그리드에 컬럼을 표시하는 방법에 대해 배워보겠습니다.

이론

컬럼에 대해 이야기 하기 전에 RealGrid의 컨셉을 이해해야 합니다. 아래 이미지는 RealGrid의 구성을 간략하게 표현한 도식 입니다.



그림에서 보는 것과 같이 RealGrid는 데이터 영역을 구현한 DataProvider와 컨트롤 영역인 Controller, 데이터 뷰 영역인 GridView로 구성되어 있습니다.
간단히 설명하면 RealGrid는 DataProvider에 들어 있는 데이터를 GridView를 통해 화면에 보여주도록 만들어 졌습니다. ItemModel이나 Controller는 GridView에 데이터가 표현되기 전에 Sorting이나 Groupping등 데이터를 조작하는 역학을 담당하고 있습니다. 각각의 구성에 대한 상세한 내용을 앞으로 차차 알아 보기로 하겠습니다.

RealGrid에서 **컬럼(Column)**이란 DataProvider에 들어있는 DataSet의 특정 필드의 값을 그리드에 표현하기 위한 부분을 말합니다. 컬럼에 대한 속성은 DataColumn를 참조하세요.

실습

1. RealGrid에 컬럼을 표시하기 위해 새로운 DataColumn 배열 객체를 생성하고 타이틀 표시를 위한 헤더 속성과 너비 속성을 입력합니다.

```
//새로운 DataColumn array 객체 생성
var columns = [
    {
        header : {
            text: "컬럼1"
        },
        width: 300
    }
];
```

2. 만들어진 DataColumn객체를 GridView의 setColumns()함수를 이용해 그리드에 입력합니다.

```
//setColumns()함수로 그리드에 반영
gridView.setColumns(columns);
```

A3 컬럼이름 바꾸기

Jun 26, 2015

RealGridJS RealGrid column

들어가며

[A2 강좌]에서는 컬럼을 만들어 보았습니다. 이번 강좌에서는 컬럼의 헤더 타이틀을 변경하는 방법에 대해 배워보겠습니다.

이론

그리드에 표시된 컬럼의 정보를 변경하기 위해서는 해당 컬럼의 헤더(header)속성을 가져온 다음 Header Title을 변경하고 다시 컬럼에 넣어주는 방법을 사용합니다. 그러면 여기서 헤더 속성을 가져오는 함수와 다시 넣어주는 함수를 알아야 겠습니다. GridView객체가 가지고 있는 두 함수는 각각 getColumnProperty() 함수와 setColumnProperty() 함수 입니다.
이 두 함수는 컬럼의 헤더 속성 뿐 아니라 데이터컬럼(DataColumn)이 가지고 있는 여러가지 속성을 수정하기 위해 사용됩니다.

실습

1. 컬럼의 헤더 속성 객체 가져오기

```
//컬럼 이름이 "col1"인 컬럼의 heder객체를 가져온다.
var header = gridView.getColumnProperty("col1", "header");
```

2. 헤더 객체의 Title속성 변경하기

```
//컬럼 헤더를 "컬럼2"로 바꾼다.
header.text = "컬럼2";
```

3. 헤더 객체 GridView에 다시 넣기

```
//헤더 객체를 GridView에 다시 넣는다.
gridView.setColumnProperty("col1", "header", header);
```

컬럼의 헤더가 "컬럼2"로 바뀐것을 확인하세요.

A4 DataProvider에 Field 만들기

Jun 26, 2015

RealGridJS RealGrid field

들어가며

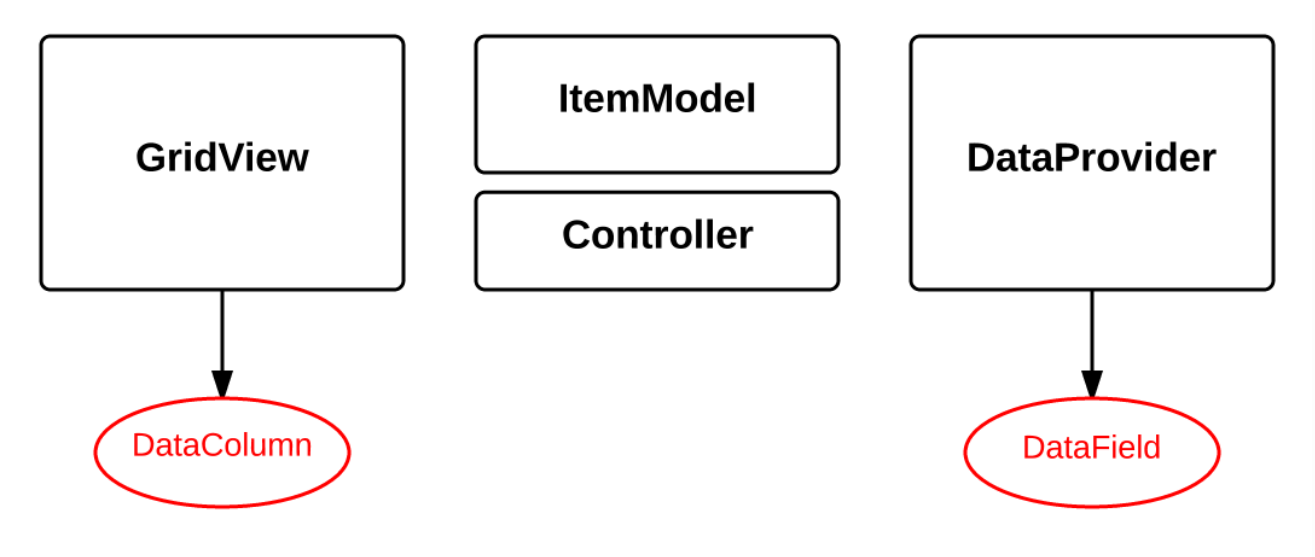
지난 강좌에서 Column을 만들어 보았지만 Field가 없는 Column은 껍데기에 불과 합니다. 이번 강좌에서는 RealGridJS의 데이터를 담고 있는 DataProvider에 Field라는 것을 만들어 데이터를 구조화 해 보기로 하겠습니다.

이론

지난 RealGridJS 컬럼 만들기에서 살펴본 RealGrid의 컨셉 이미지에 Field와 Column이 속한 영역을 표시해 보았습니다. 그림에서 Column이 `속한 영역`은 GridView이며 Field가 속한 영역은 DataProvider입니다.

DataProvider는 원본 데이터를 관리하기 위한 클래스이며 하위 클래스인 LocalDataProvider와 TreeDataProvider는 각각 그리드와 트리그리드의 데이터관리를 위한 클래스입니다.

> `속한영역`이라고 표현 하는 것이 조금 어색하긴 하지만 그렇게 표현하겠습니다.



RealGrid에서
Field는 DataField 라는 클래스로 정의 되어 있으며,
Column은 DataColumn 이라는 클래스로 정의 되어 있습니다.

실습

1. DataField객체의 배열을 생성하여 fields변수에 넣습니다.

```
//새로운 DataField 배열 객체 생성
var fields = [
    {
        fieldName: "field1"
    },
    {
        fieldName: "field2"
    }
];
```

2. 생성한 DataField 배열을 DataProvider의 setFields() 함수의 인자로 넘겨 필드셋을 구성합니다.

```
//setColumns()함수로 그리드에 반영
dataProvider.setFields(fields);
```

A5 컬럼-필드 연결하기

Jun 29, 2015

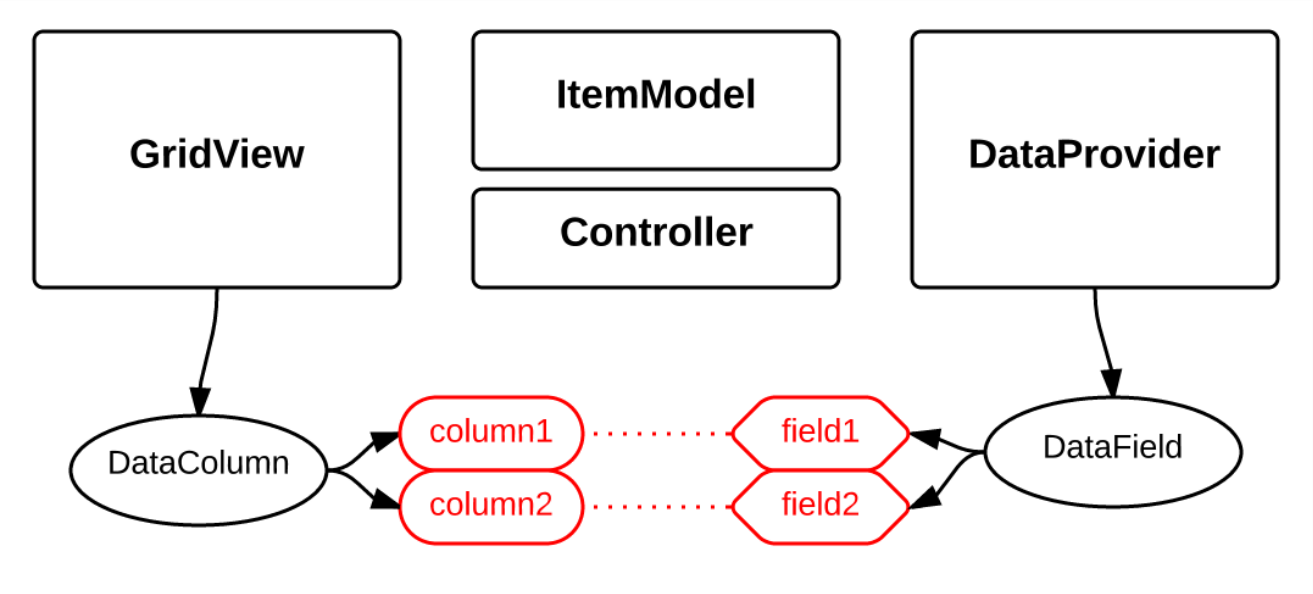
RealGridJS RealGrid column field

들어가며

필드와 컬럼을 연결한다는 표현은 조금 이상 합니다. 하지만 그냥 그렇게 표현하겠습니다. DataProvider에 들어 있는 데이터를 GridView에 표현 하는 것은 RealGrid의 중요한 기능 중 하나입니다.
이 기능을 구현하기 위해 필드와 컬럼을 연결 해야 합니다.

이론

그림을 보겠습니다.
필드는 `Data`영역이고 컬럼은 `View`영역입니다.



그림에서 처럼 하나의 필드는 하나의 컬럼과 연결 가능합니다. RealGrid는 하나의 필드에 여러개의 컬럼을 연결 할 수도 있습니다. 이것은 데이터를 다양한 방법으로 표현 가능한 RealGrid의 장점이기도 합니다. 하나의 필드에 여러 개의 컬럼을 연결해 보는 연습은 [A7 강좌]에서 배울 수 있습니다.

필드와 컬럼을 연결하는 방법은 간단합니다. 컬럼 객체를 생성할때 fieldName속성에 연결할 필드의 이름을 지정 하면 됩니다.

fieldName속성은 데이터컬럼(DataColumn)페이지를 참조하세요.

실습

1. 두 개의 필드를 가진 배열 객체를 생성하고 DataProvider에 입력 합니다.

```
//두 개의 필드를 가진 배열 객체를 생성합니다.
var fields = [
    {
        fieldName: "field1"
    },
    {
        fieldName: "field2"
    }
];
//DataProvider의 setFields함수로 필드를 입력합니다.
dataProvider.setFields(fields);
```

2. field1과 연결된 컬럼을 생성하고 GridView에 입력합니다.

```
//필드와 연결된 컬럼을 가진 배열 객체를 생성합니다.
var columns = [
    {
        name: "col1",
        fieldName: "field1",
        header : {
            text: "컬럼1"
        },
        width: 200
    },
    {
        name: "col2",
        fieldName: "field2",
        header : {
            text: "컬럼2"
        },
        width: 200
    }
];
//컬럼을 GridView에 입력 합니다.
gridView.setColumns(columns);
```

A6 RealGrid에 데이터 넣기

Jun 30, 2015

RealGridJS RealGrid column field dataprovider setRows

들어가며

이번 강좌는 [A5 강좌]에서 이어지는 내용입니다.
그리드에 필드와 컬럼이 연결된 상태에서 DataProvider에 데이터를 넣는 방법을 배워 보겠습니다.

이론

DataProvider에 데이터를 넣는 방법은 아래와 같이 여러가지 방법이 있습니다. 이 강좌에서는 LocalDataProvider.setRows()함수를 사용합니다.

실습

1. 데이터행 배열 객체를 생성하고 데이터를 입력 합니다.

```
var data = [
    ["data1", "data2", "data3"]
];
```

2. setRows()함수로 LocalDataProvider에 원본 데이터를 입력합니다.

```
    dataProvider.setRows(data);
```

A7 필드 하나에 컬럼 두 개 연결하기

Jun 30, 2015

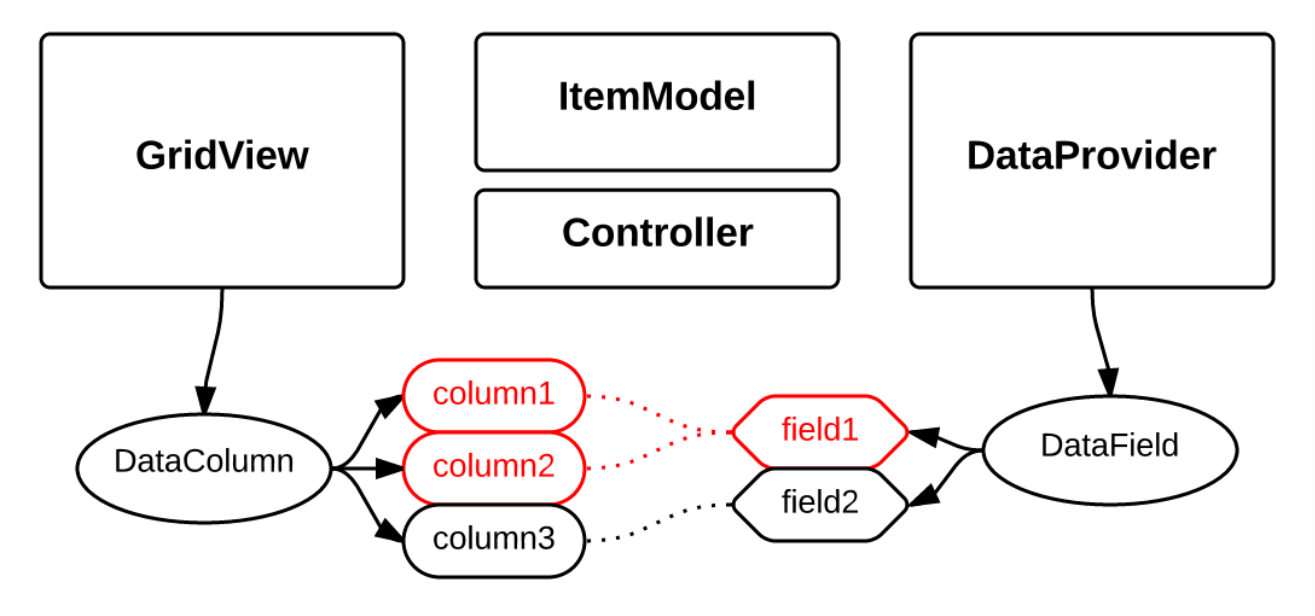
RealGridJS RealGrid column field dataprovider

들어가며

이 강좌에서는 [A6 강좌]와는 달리 하나의 필드를 두 개의 컬럼과 연결하는 방법을 배웁니다.

이론

아래 그림에서 처럼 필드와 컬럼의 관계는 1:n의 관계 입니다. 이 방법은 그리드를 활용한 업무에서 다양한 데이터 표현기법을 가능 하게 해 줍니다.
예를 들어 하나의 필드 데이터를 숫자컬럼과 바셀렌더러(BarCellRenderer)컬럼에 연결하면 시각적인 데이터표현 효과를 높일 수 있습니다.



실습

1. 두 번째 컬럼의 fieldName 속성에 “field1”을 연결합니다.

```
var columns = [
    {
        name: "col1",
        fieldName: "field1",
        header : {
            text: "컬럼1"
        },
        width: 150
    },
    {
        name: "col2",
        fieldName: "field1",
        header : {
            text: "컬럼2"
        },
        width: 150
    }
];
```

2. 데이터행 배열 객체를 생성하고, setRows()함수로 LocalDataProvider에 원본 데이터를 입력합니다.

```
    var data = [
        ["data1", "data2"]
    ];
    dataProvider.setRows(data);
```

A8 포커스셀(Focused Cell) 이해하기

Jul 6, 2015

RealGridJS RealGrid dataprovider focus

들어가며

이번 강좌에서는 그리드의 포커스셀(Focused Cell)에 대해 알아보고 포커스셀의 정보를 가져오거나 특정셀로 포커스를 이동시켜보겠습니다.

이론

RealGrid에서 포커스셀이란 커서가 위치한 셀을 의미합니다. 포커스셀의 정보를 가져오려면 GridBase.getCurrent()를 이용하고 포커스셀의 정보를 입력하려면 GridBase.setCurrent()를 이용합니다. 두 함수의 인자와 리턴값인 CellIndex는 데이터셀(DataCell)의 정보를 관리하기 위한 클래스입니다. CellIndex의 속성에 대한 정보는 다음 강좌에서 배울수 있습니다.

실습

1. getCurrent()함수로 현재 포커스된 셀의 CellIndex를 가져옵니다. 가져온 CellIndex객체에 dataRow, column, fieldName속성을 변경하여 포커스셀의 위치를 이동합니다.

```
$("#btnToggleFocus").on("click", function(){
    //현재 포커스된 셀정보 가져오기
    var focusCell = gridView.getCurrent();
    //이동할 행 번호(RowId), 컬럼정보 입력
    focusCell.dataRow = 0;
    if (focusCell.fieldName == "field1") {
        focusCell.column = "col2";
        focusCell.fieldName = "field2";
    }
    else {
        focusCell.column = "col1";
        focusCell.fieldName = "field1";
    }
    //포커스된 셀 변경
    gridView.setCurrent(focusCell);
})
```

A9 여러행 데이터와 RowId 이해하기

Jul 6, 2015

RealGridJS RealGrid dataprovider RowId dataRow CellIndex

들어가며

이번 강좌에서는 여러행의 데이터를 그리드에 입력하는 방법과 RowId의 개념을 배워보겠습니다.

이론

여러행의 데이터를 입력하는 방법이 따로 있는 것은 아닙니다. 기본적으로 DataProvider.setRows()함수에 인자로 데이터 배열 객체를 넣어 주면 됩니다.
사실 이번 강좌에서 이해해야 할 부분은 DataProvider에 입력된 데이터는 고유의 행 번호를 가지게 된다는 것과 이 행 번호를 `RowId`라고 부른다는 것입니다. RowId는 DataProvider에 들어있는 데이터세트(DataSet)의 각 행에 부여된 고유번호입니다.
RowId는 A8 강좌에서 알아본 CellIndex클래스의 `dataRow`속성을 통해 확인할 수 있습니다.

실습

1. 여러행 데이터를 배열로 만들어 data변수에 넣습니다. DataProvider.setRows()를 호출 하면서 인자로 넘겨줍니다.

```
var data = [
    ["data1-1", "data1-2"],
    ["data2-1", "data2-2"],
    ["data3-1", "data3-2"],
    ["data4-1", "data4-2"]
];
dataProvider.setRows(data);
```

2. 버튼을 클릭하면 GridView.getCurrent()로 가져온 CellIndex정보에서 dataRow속성값을 팝업 합니다.

```
$("#btnPopupRowId").on("click", function(){
    var focusCell = gridView.getCurrent();
    alert(focusCell.dataRow);
});
```

A10 ItemModel과 ItemIndex 이해하기

Jul 9, 2015

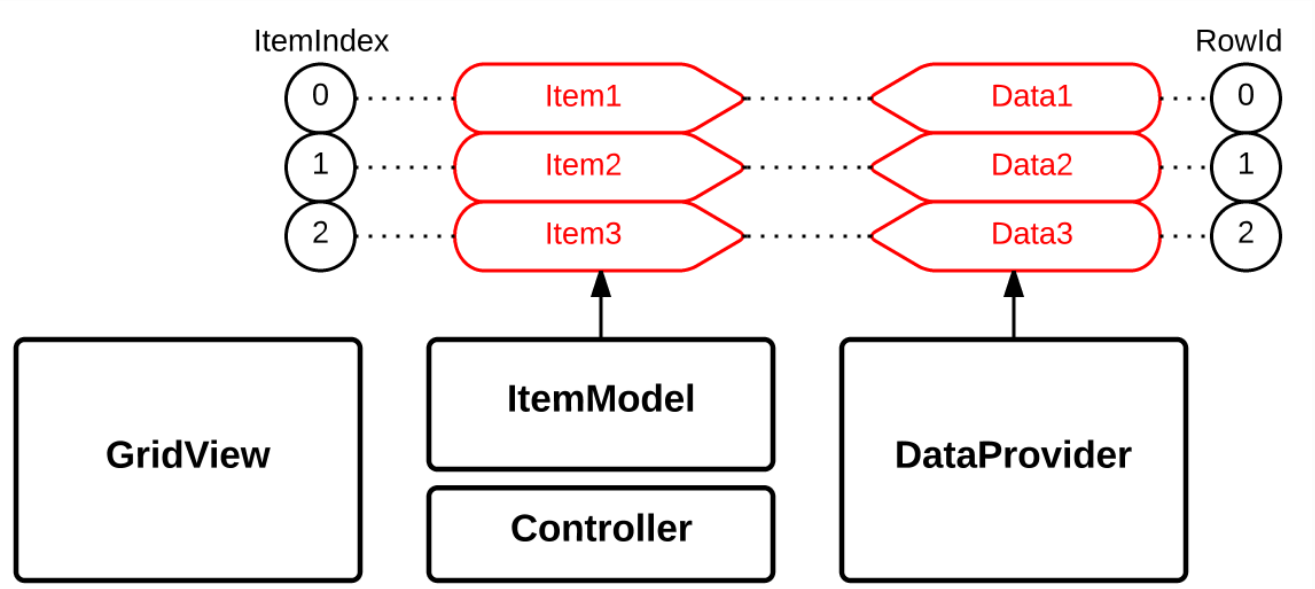
RealGridJS RealGrid dataprovider ItemModel itemIndex CellIndex

들어가며

A9 강좌에서 `RowId`에 대해 배워보았습니다. 이번 강좌에서는 그와 유사한 개념인 `ItemIndex`에 대해 배워보겠습니다.

이론

기본적으로 아래 그림과 같이 Item과 Data는 1:1 매핑이 되어 있습니다. 하지만, 소팅(Sorting), 그룹핑(Grouping)등 컨트롤러(Controller)의 다양한 기능을 그리드에 표현하기 위해 더 많은 Item들이 생겨납니다. 이런 기능은 앞으로 남은 많은 강좌들에서 배울수 있습니다.



`ItemIndex`는 Item의 고유한 행 번호 입니다. itemIndex는 A8 강좌에서 알아본 CellIndex클래스의 `itemIndex`속성을 통해 확인 할 수 있습니다.

실습을 통해 현재 포커스된 셀의 ItemIndex를 팝업해 보겠습니다.

실습

소스코드는 A9 강좌와 거의 동일합니다.

1. 버튼을 클릭하면 GridView.getCurrent()로 가져온 CellIndex.itemIndex를 팝업합니다.

```
$("#btnPopupItemIndex").on("click", function(){
    var focusCell = gridView.getCurrent();
    alert(focusCell.itemIndex);
});
```

A11 Data와 Item의 다른점은?

Jul 9, 2015

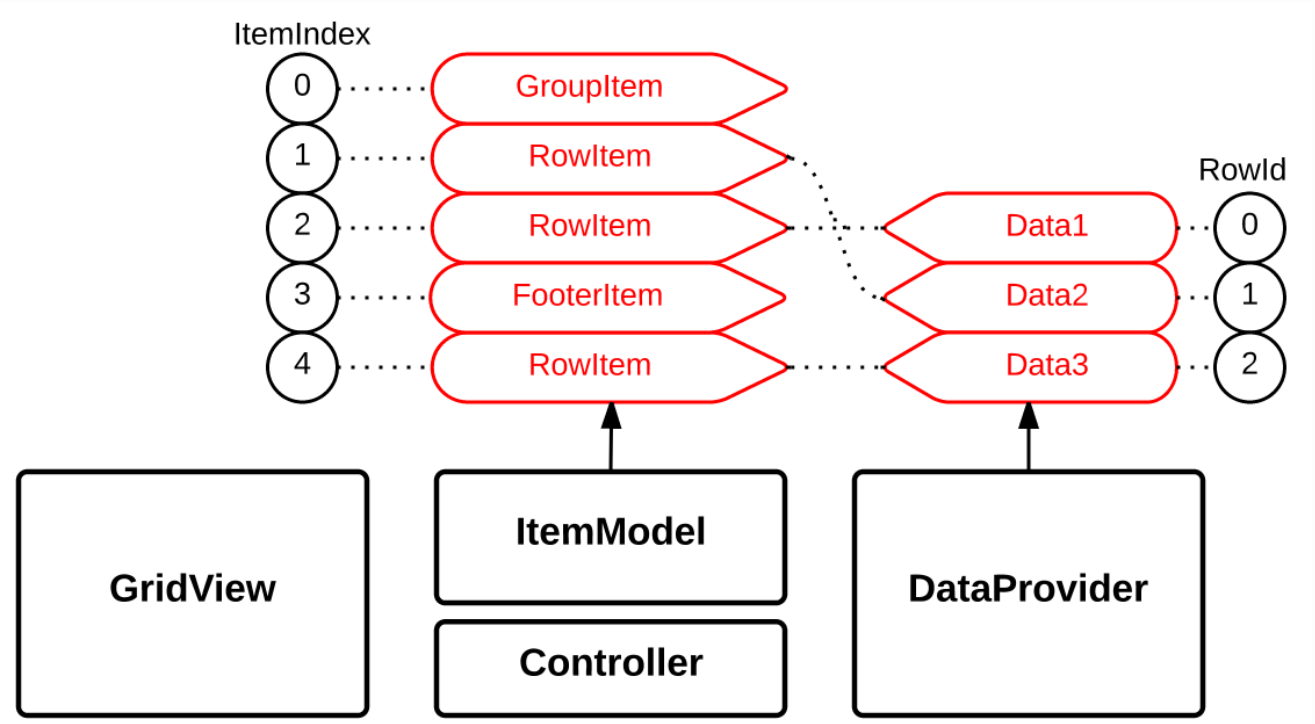
RealGridJS RealGrid dataprovider itemmodel itemid datarow rowid

들어가며

A9 강좌와 A10 강좌에서 `RowId`와 `ItemIndex`에 대해 배워보았습니다. 이번 강좌에서는 Data와 Item에 대해 알아보겠습니다.

이론

Data와 Item은 RealGrid의 공식적인 명칭이 아니지만 튜토리얼에서는 이해를 돕기 위해 DataProvider와 ItemModel의 행(Row)정보를 각각 Data와 Item으로 표현하겠습니다.
아래 그림은 Data와 Item의 관계를 표현하고 있습니다. 동시에, Item의 종류도 설명합니다.



- Item의 순서는 Data의 순서는 동일할 수도 있고, 아닐 수도 있습니다.
- Item은 "row", "group", "footer", "tree"의 형식으로 구분됩니다.
- Item에 대한 자세한 설명은 Grid Item을 참조하세요.

Item과 Data의 개념은 앞으로 남은 많은 강좌들을 실습해 가면서 좀더 깊이 이해할 수 있습니다.
이번 강좌에서는 ItemIndex와 RowId가 서로 다를수 있다는 사실을 정렬(Sorting)기능을 통해 확인해 보겠습니다.
RealGrid에서 정렬은 Data를 정렬하는 것이 아니고 Item을 정렬하는 것입니다. 즉, 정렬순서가 변경되면 RowId와는 관계가 없고, ItemIndex에 변동이 발생합니다. 이런 사실을 실습을 통해 확인해 보겠습니다.

실습

소스코드는 A10 강좌에서 필드와 컬럼, 데이터는 모두 동일합니다.

1. setCurrent2Row()함수는 포커스를 두 번째 행에 가도록 하는 함수입니다. GridView.getCurrent()함수의 리턴값인 CollIndex의 dataRow값을 변경하도록 했지만, 현재상태는 ItemIndex와 RowId가 같기 때문에 코드에서 `current.dataRow = 1;`을 `current.itemIndex = 1;`로 바꾸어도 동일한 결과를 얻을 수 있습니다.

```
//동적으로 두 번째 행에 포커스 되도록 함수
function setCurrent2Row(){
    var current = {};
    current.dataRow = 1;
    gridView.setCurrent(current);
}
```

2. popupIndex()함수는 포커스된 셀의 ItemIndex와 RowId를 팝업합니다.

```
//ItemIndex와 RowId를 팝업하는 함수
function popupIndex(){
    var focusCell = gridView.getCurrent();
    alert("ItemIndex:" + focusCell.itemIndex + ", RowId:" + focusCell.dataRow);
}
```

3. changeOrder()함수는 두 번째 컬럼 데이터의 역순으로 그리드의 정렬순서를 변경합니다. GridView.orderBy();함수에 대해서는 별도의 강좌에서 배울수 있습니다.

```
//두 번째 필드의 데이터를 기준으로 역순으로 정렬
function changeOrder(){
    var fields = ["field2"];
    var dirs = [RealGridJS.SortDirection.DESCENDING];
    gridView.orderBy(fields, dirs);
}
```

4. 아래 코드는 각 버튼의 클릭 이벤트 입니다.

```
//버튼을 클릭하면 포커스행을 두번째 행으로 선택합니다.
$("#btnSetCurrent2Row").on("click", function(){
    setCurrent2Row();
})

//버튼을 클릭하면 ItemIndex와 RowId를 표시 합니다.
$("#btnPopupIndex1").on("click", function(){
    popupIndex();
})
//버튼을 클릭하면 정렬순서를 바꿉니다.
$("#btnChangeOrder").on("click", function(){
    changeOrder();
})

//버튼을 클릭하면 ItemIndex와 RowId를 표시 합니다.
$("#btnPopupIndex2").on("click", function(){
    popupIndex();
})
```

A12 소팅(sorting), 데이터 정렬하기 - I 단일 컬럼 정렬

Jul 13, 2015

RealGridJS RealGrid dataprovider itemmodel sort sorting exclusive

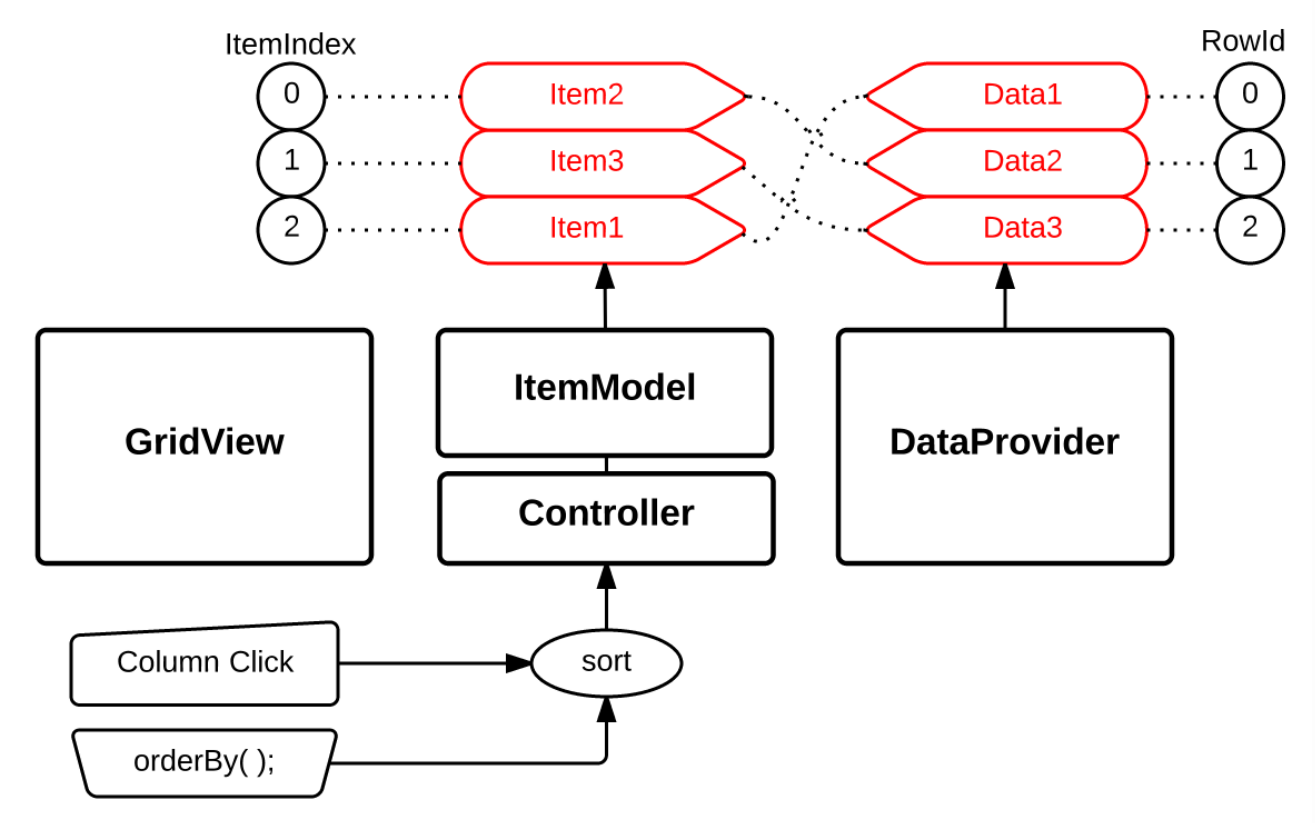
들어가며

[A11 강좌]에서 이미 `GridBase.orderBy();` 함수를 이용해 정렬하는 방법이 언급되었습니다. [A12 강좌], [A13 강좌], [A14 강좌]에서 정렬(sorting)에 대해 좀더 자세히 배워보겠습니다.

이론

> RealGrid에서 `정렬`이란 특정 컬럼의 데이터값에 의해 행의 순서를 재배열 하는 것을 의미합니다.

그림에서 처럼 정렬 명령은 Controller가 받아서 ItemModel을 재구성하는 과정으로 처리됩니다. 정렬 명령을 내리는 방법은 `컬럼 클릭(Column Click)` 방법과 `orderBy();`함수를 호출 하는 방법이 있습니다.



[그림 12-1]

- **SortStyle**
  정렬은 **단일 컬럼 정렬(exclusive)**과 **다중 컬럼 정렬(inclusive, reverse)**로 SortStyle을 구분 합니다. SortStyle은 SortingOptions클래스를 통해 그리드에 적용 할 수 있습니다.
- **SortDirection**

정렬 방향은 **순방향(ascending)**과 **역방향(descending)**의 두 가지가 있습니다. 순방향이란 작은값에서 큰값의 순서로 정렬하는 것이며, 역방향은 그 반대의 순서로 정렬하는 것입니다.
정렬방향은 SortDirection의 이름으로 정의 되어 있습니다. SortDirection은 GridView.orderBy()함수의 인자로 넣어 각 필드의 정렬 방향을 결정 합니다.

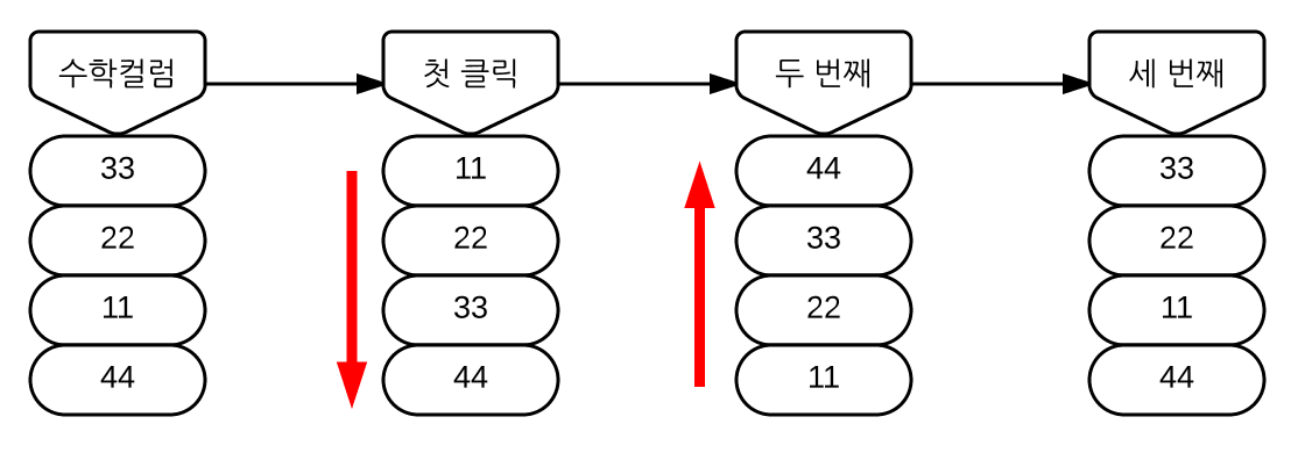
이번 강좌에서는 정렬을 위해 그리드에 옵션을 설정하는 방법과 컬럼 클릭 정렬방법중 단일 컬럼 정렬(exclusive)에 대해 배워보겠습니다.

컬럼 클릭에 의한 정렬은 아래 세가지 방법으로 토글(toggle) 됩니다.

- 첫 번째 클릭: 순방향(ascending) 정렬
- 두 번째 클릭: 역방향(descending) 정렬
- 세 번째 클릭: 원래 데이터의 순서(RowId 순서)로 정렬

단일 컬럼 정렬

아래 그림은 실행화면에서 수학컬럼을 클릭할 경우 정렬이 실행되는 순서를 표시한 그림입니다. 실제 RealGrid에서 아래 그림과 같이 클릭했을 경우 정렬되는지 확인해 보세요.
단일 컬럼 정렬인 경우 정렬된 컬럼이 아닌 다른 컬럼을 클릭 할 경우 이전 컬럼의 정렬 순서는 모두 무시하고 새롭게 클릭된 컬럼으로 다시 정렬 합니다.



[그림 12-2]

실습

1. 그리드의 데이터를 위한 코드는 지금까지 연습했던 코드와 같습니다. 세개의 필드와 세개의 컬럼, 그리고 4개 행의 데이터를 입력하는 코드입니다.

```
//세 개의 필드를 가진 배열 객체를 생성합니다.
var fields = [
    {
        fieldName: "field1"
    },
    {
        fieldName: "field2"
    },
    {
        fieldName: "field3"
    }
];
//DataProvider의 setFields함수로 필드를 입력합니다.
dataProvider.setFields(fields);

//필드와 연결된 컬럼을 가진 배열 객체를 생성합니다.
var columns = [
    {
        name: "col1",
        fieldName: "field1",
        header : {
            text: "이름"
        },
        width: 150
    },
    {
        name: "col2",
        fieldName: "field2",
        header : {
            text: "국어"
        },
        width: 150
    },
    {
        name: "col3",
        fieldName: "field3",
        header : {
            text: "수학"
        },
        width: 150
    }
];
//컬럼을 GridView에 입력 합니다.
gridView.setColumns(columns);

var data = [
    ["송윤아", "10", "33"],
    ["전도연", "20", "22"],
    ["하지원", "20", "11"],
    ["전지현", "10", "44"]
];
dataProvider.setRows(data);
```

2. setSortStyles()함수에서는 정렬 옵션을 설정하기 위해 GridBase.setSortOptions();함수를 호출 합니

다.

```
//그리드의 소트 옵션 설정 함수
function setSortStyles(style) {
    var options = {};
    options.style = style;
    gridView.setSortingOptions(options);
}
```

3. SortStyle을 NONE과 EXCLUSIVE로 설정하기 위해 버튼클릭 이벤트를 작성 합니다.

```
// SortStyle을 NONE으로 설정
$("#btnChangeSortStyleNone").on("click", function(){
    setSortStyles(RealGridJS.SortStyle.NONE);
})

// SortStyle을 EXCLUSIVE로 설정
$("#btnChangeSortStyleExclusive").on("click", function(){
    setSortStyles(RealGridJS.SortStyle.EXCLUSIVE);
})
```

A13 소팅(sorting), 데이터 정렬하기 - II 다중 컬럼 정렬

Jul 14, 2015

RealGridJS RealGrid dataprovider itemmodel sort sorting inclusive revers

들어가며

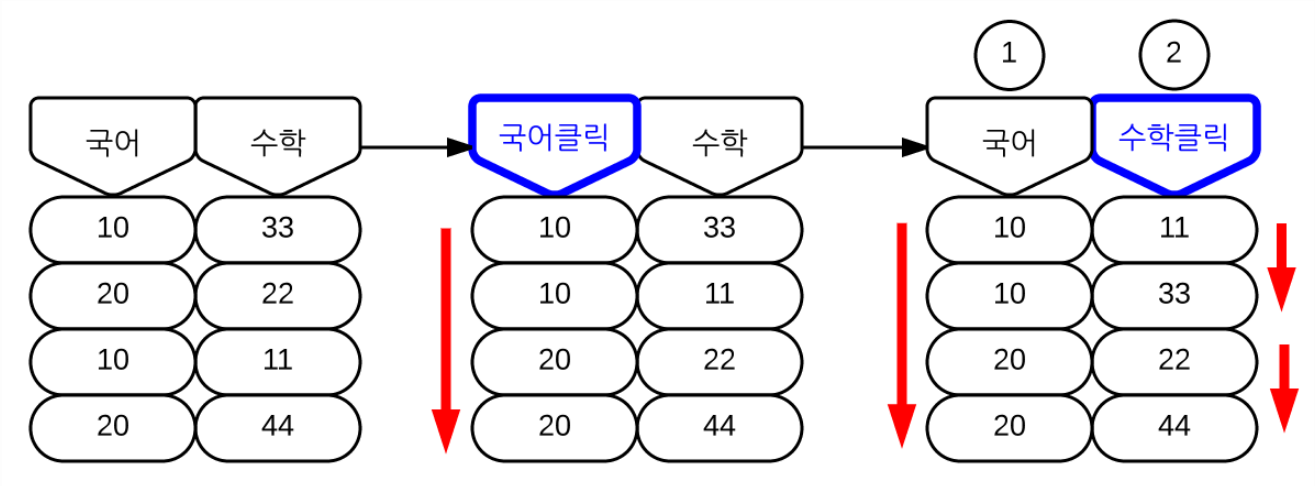
지난 [A12 강좌]에서 RealGrid의 정렬에 대한 기본적인 개념과 컬럼 클릭을 통한 단일 컬럼 정렬 방법에 대해 배웠습니다. 이번 강좌에서는 다중 컬럼 정렬에 대해 배워보겠습니다.

이론

다중 컬럼 정렬에 대한 옵션은 INCLUSIVE와 REVERS가 있습니다. INCLUSIVE는 먼저 클릭한 컬럼을 우선 정렬하고 REVERS는 나중에 클릭한 컬럼을 우선 정렬합니다. INCLUSIVE와 REVERS를 아래 그림을 보면서 조금 더 설명하겠습니다.

INCLUSIVE

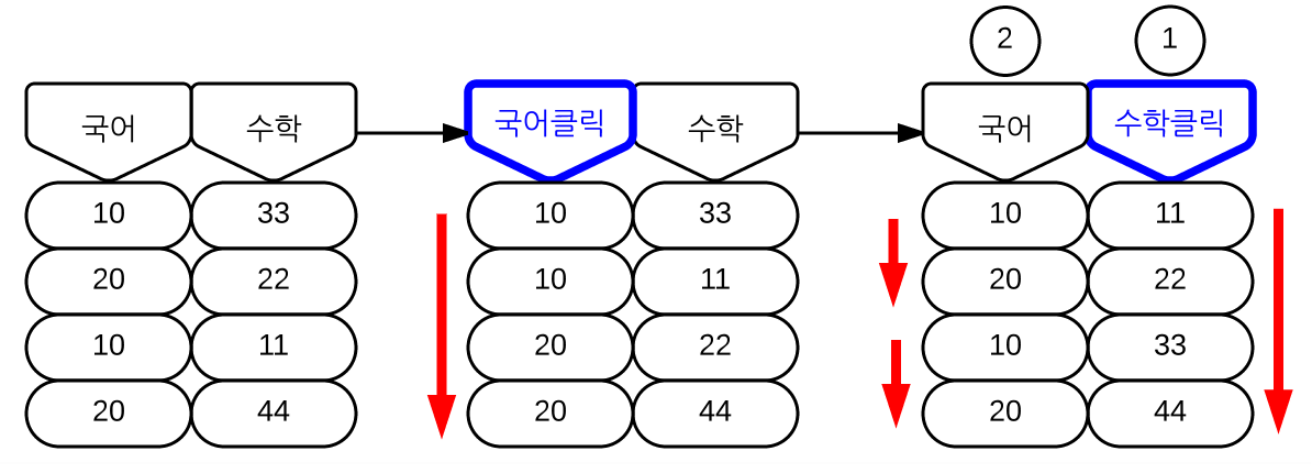
SortStyle을 INCLUSIVE로 설정한 경우
먼저 국어 컬럼을 클릭하고 다음에 수학 컬럼을 클릭해 보겠습니다. 그러면 국어 컬럼의 정렬은 고정된채 수학 컬럼만 다시 정렬합니다. 그림에서 원 안에 숫자는 정렬 우선순위를 의미 합니다.



[그림 13-1]

REVERS

SortStyle을 REVERS로 설정한 경우
먼저 국어 컬럼 클릭 -> 수학 컬럼 클릭해 보겠습니다. 이번에는 나중에 클릭한 수학 컬럼으로 우선 정렬하고 수학 컬럼의 정렬 순위가 두 번째로 밀린 것을 확인 할 수 있습니다.



[그림 13-2]

REVERS는 특별한 경우가 아니면 사용할 필요 없습니다.

INCLUSIVE와 REVERS, 두 가지 옵션이 있지만, 사실 둘중 하나의 옵션만을 사용해도 동일하게 정렬된 결과를 만들어낼 수 있습니다.
위의 그림을 예로들면, SortStyle을 INCLUSIVE로 놓고 수학 컬럼 -> 국어 컬럼의 순서로 클릭하면 REVERS로 국어 컬럼 -> 수학 컬럼의 순서로 클릭한 결과와 동일한 결과를 얻을 수 있습니다.

이 내용은 아래 실습을 통해 직접 테스트해 보세요.

실습

1. 그리드의 정렬상태를 초기화 하기 위해 GridBase.orderBy();함수에 빈 배열 값을 넘겨주는 resetOrders()함수를 만듭니다.

```
//정렬 순서를 초기화하는 함수
function resetOrders() {
    gridView.orderBy([], []);
}

//그리드의 소트 옵션 설정 함수
function setSortStyles(style) {
    var options = {};
    options.style = style;
    gridView.setSortingOptions(options);

    resetOrders();
}
```

2. SortStyle을 INCLUSIVE와 REVERSE로 설정하기 위해 버튼 클릭 이벤트를 작성 합니다.

```
// SortStyle을 INCLUSIVE으로 설정
$("#btnChangeSortStyleInclusive").on("click", function(){
    setSortStyles(RealGridJS.SortStyle.INCLUSIVE);
})

// SortStyle을 REVERSE로 설정
$("#btnChangeSortStyleReverse").on("click", function(){
    setSortStyles(RealGridJS.SortStyle.REVERSE);
})
```

A14 소팅(sorting), 데이터 정렬하기 - III orderBy()함수 사용하기

Jul 15, 2015

RealGridJS RealGrid dataprovider itemmodel sort sorting orderby

들어가며

지난 [A12 강좌], [A13 강좌]에서는 컬럼을 클릭하여 데이터를 정렬하는 방법에 대해 알아보았습니다. 이번 강좌에서는 GridBase.orderBy()함수를 이용해 정렬하는 방법에 대해 배워보겠습니다.

이론

orderBy함수의 인자는 FieldName의 배열값과 SortDirection의 배열값입니다. FieldName와 SortDirection은 짝이 맞아야 하며 각각의 짝에 해당하는 필드와 방향으로 정렬을 실행 합니다. 정렬상태를 완전히 초기화 하기 위해서는 아래 코드처럼 FieldName배열과 SortDirection배열, 두 인자에 모두 빈값을 넘겨주면 됩니다.

```
    //정렬 순서 초기화하기
    gridView.orderBy([], []);
```

동시에 여러 컬럼을 한 번에 정렬할 필요가 있는 경우 사용할 수 있습니다.

실습

1. 그리드의 정렬상태를 초기화 하기 위해 GridBase.orderBy();함수에 빈 배열 값을 넘겨주는 resetOrders()함수를 만듭니다.

```
//정렬 순서를 초기화하는 함수
function resetOrders() {
    gridView.orderBy([], []);
}
```

2. btnChangeOrders INCLUSIVE와 REVERSE로 설정하기 위해 버튼 클릭 이벤트를 작성 합니다.

```
// SortStyle을 INCLUSIVE으로 설정
$("#btnChangeSortStyleInclusive").on("click", function(){
    setSortStyles(RealGridJS.SortStyle.INCLUSIVE);
})

// SortStyle을 REVERSE로 설정
$("#btnChangeSortStyleReverse").on("click", function(){
    setSortStyles(RealGridJS.SortStyle.REVERSE);
})
```

3. orderBy()함수를 사용하여 국어컬럼, 수학컬럼을 순방향 정렬하는 코드를 작성합니다.

```
// 국어, 수학 컬럼을 순방향 정렬하기위해 orderBy() 함수 사용
$("#btnChangeOrders").on("click", function(){
    gridView.orderBy(["field2", "field3"], ["ascending", "ascending"]);
})
```

A15 로우 그룹핑(row grouping) - I 드래깅(dragging)을 이용하여 그룹핑

Jul 15, 2015

RealGridJS RealGrid dataprovider itemmodel group grouping groupby

들어가며

[A15 강좌], [A16 강좌]에서는 로우 그룹핑(row grouping)에 대해 배워보겠습니다.

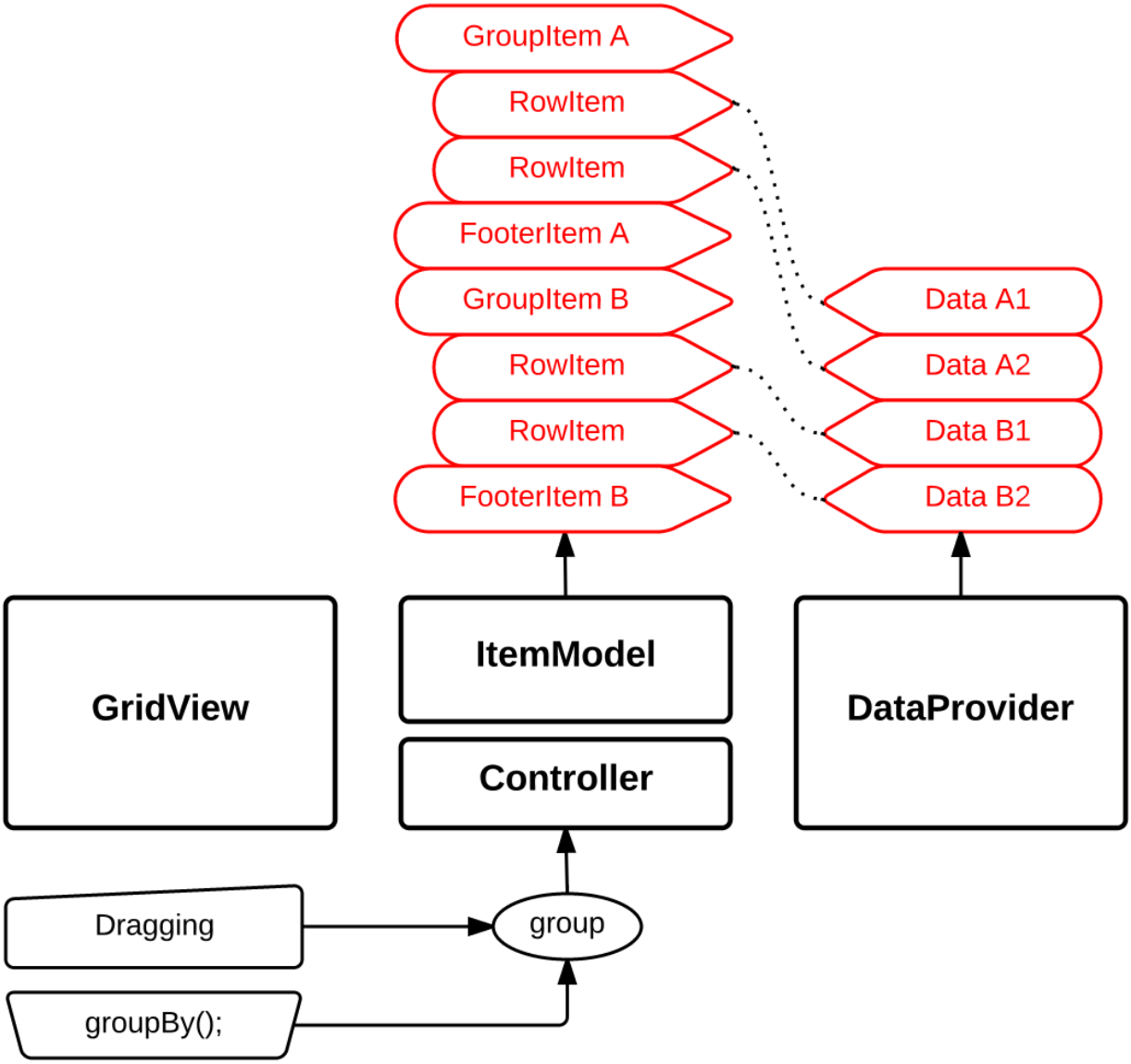
이 강좌를 연습하기 전에 RealGrid의 컴포넌트를 반드시 이해해야 합니다. 링크로 이동하여 RealGrid 컴포넌트의 영역과 명칭을 확인하세요.

이론

RealGrid에서 그룹핑이란 정해진 조건으로 데이터를 분류하여 묶는 것을 의미 합니다. 그룹핑은 로우 그룹핑(row grouping)과 컬럼 그룹핑(column grouping)으로 구분됩니다.

이번 강좌와 다음 강좌에서 배울 로우 그룹핑(row grouping)이란 특정 컬럼을 기준으로 그룹핑하는 것을 의미합니다. 즉, 해당 컬럼의 데이터값이 동일한 행을 묶는 것을 의미합니다.

[A11 강좌]에서 우리는 ItemModel이 가진 아이템의 종류(ItemType)에 대해 배웠습니다. 로우 그룹핑을 할 때 ItemModel에는, 로우아이템(row item)과는 별도로 그룹아이템(group item)과 푸터아이템(footer item)이 만들어 집니다.
아래 그림을 보면 ItemModel에 그룹아이템과 푸터아이템이 어떤형태로 존재하게 되는지 이해 할 수 있습니다.
[A11 강좌]에 의하면 그룹아이템과 푸터아이템도 ItemIndex를 갖는 별도의 행으로 그리드에 표현됩니다. 이 두 아이템은 각각 그룹핑된 아이템의 헤더와 푸터 역할을 하는 아이템이며 로우 그룹 헤더(row group header)는 그룹의 제목(title)이나 갯수(count)를 표현 할 수 있고, 로우 그룹 푸터(row group footer)는 컬럼의 합계(sum)나 평균(avg)등 그룹핑된 데이터 집합에 대한 계산된 값을 표현 할 수 있는 영역입니다.



[그림 A15-1]

> 호기심 많은 분들을 위해 `GridBase.getModelAs(itemIndex);`라는 함수를 소개해 드립니다.
> 로우 그룹핑을 연습한 다음, 그리드에서 각각의 행이 어떤 `ItemType`인지 확인해 보세요.

```
<script>
console.log(gridView.getModelAs(0));
</script>

CONSOLE >
Object
 $_child: 1
 checked: false
 count: 4
 dataRow: -1
 expanded: true
 footerId: 14
 id: 13
 itemIndex: 0
 parentId: NaN
 type: "group"
 __proto__: Object
```

로우 그룹핑

위 [그림 A15-1]을 보면 Controller에 그룹핑 명령을 내리는 방법은 `Dragging` 방법과 `groupBy();` 함수를 사용하는 방법이 있다는 것을 알 수 있습니다. 이번 강좌에서는 패널(panel)영역으로 컬럼 헤더(column header)를 `Dragging` 하여 로우 그룹핑하는 방법에 대해 배워보고 연습해 보겠습니다.

중첩 로우 그룹핑

로우 그룹핑은 여러개의 컬럼을 동시에 중첩하여 그룹핑 할 수 있습니다. 그룹핑할 컬럼을 패널에 드래깅하는 순서대로 중첩 그룹핑이 적용됩니다.

로우 그룹핑 해제

로우 그룹핑을 해제하려면 패널에 있는 컬럼 헤더를 패널영역 밖으로 드래깅 하면 됩니다.

로우 그룹핑에 의한 소팅

로우 그룹핑을 실행하면 그룹핑된 컬럼을 자동으로 순방향(assending)으로 정렬됩니다. 중첩 그룹핑시에도 마찬가지로 상위 컬럼의 범위내에서 하위 컬럼이 순방향으로 정렬됩니다.

실습

1. 우선 그룹핑을 연습하기 위해 데이터를 조금 더 확장 했습니다. 직업과 성별 컬럼을 추가하고 몇개의 데이터를 추가 했습니다.

```
//다섯개의 필드를 가진 배열 객체를 생성합니다.
var fields = [
    {
        fieldName: "field1"
    },
    {
        fieldName: "field2"
    },
```

```
    {
        fieldName: "field3"
    },
    {
        fieldName: "field4"
    },
    {
        fieldName: "field5"
    }
];
//DataProvider의 setFields함수로 필드를 입력합니다.
dataProvider.setFields(fields);

//필드와 연결된 컬럼 배열 객체를 생성합니다.
var columns = [
    {
        name: "col1",
        fieldName: "field1",
        header : {
            text: "직업"
        },
        width: 100
    },
    {
        name: "col2",
        fieldName: "field2",
        header : {
            text: "성별"
        },
        width: 100
    },
    {
        name: "col3",
        fieldName: "field3",
        header : {
            text: "이름"
        },
        width: 100
    },
    {
        name: "col4",
        fieldName: "field4",
        header : {
            text: "국어"
        },
        width: 100
    },
    {
        name: "col5",
        fieldName: "field5",
        header : {
            text: "수학"
        },
        width: 100
    }
];
//컬럼을 GridView에 입력 합니다.
```

```
gridView.setColumns(columns);

var data = [
    ["배우", "여자", "송윤아", "10", "33"],
    ["배우", "여자", "전도연", "20", "22"],
    ["가수", "여자", "이선희", "40", "33"],
    ["배우", "여자", "하지원", "10", "11"],
    ["가수", "여자", "소찬휘", "30", "55"],
    ["가수", "여자", "박정현", "40", "22"],
    ["배우", "여자", "전지현", "20", "44"]
];
dataProvider.setRows(data);
```

A16 로우 그룹핑(row grouping) - II groupBy()함수로 그룹핑

Jul 15, 2015

RealGridJS RealGrid dataprovider itemmodel group grouping groupby

들어가며

[A15 강좌]에서는 로우 그룹핑(row grouping)의 개념과 컬럼 헤더를 드래깅하여 그룹핑하는 방법에 대해 배워보았습니다. 이번 강좌에서는 [A15 강좌]와 동일한 결과를 GridView.groupBy()함수를 이용해 구현하는 방법을 배워보겠습니다.

이론

GridView.groupBy()함수는 그룹핑할 필드들을 배열형식의 인자로 넘겨줍니다. 인자로 넘겨지는 필드가 여러개인 경우 필드의 순서에 따라 차례로 중첩하여 그룹핑됩니다.

groupBy()함수가 호출 될 때마다 기존의 그룹핑 정보는 모두 초기화 된다는 사실에 주의해야 합니다.
모든 그룹상태를 초기화 하기 위해서는 `groupBy([]);` 와 같이 빈 배열값을 인자로 넘겨주면 됩니다.

groupBy()함수를 호출하여 그룹핑하면 기본적으로, 인자로 넘겨진 필드로 순방향 정렬됩니다. 아래 실습에서 각각의 버튼을 클릭할때 해당 컬럼으로 **순방향 정렬(sort by ascending)**되는 모습을 확인하세요.

실습

1. 실습을 위한 기본코드는 [A15 강좌]와 동일합니다.
2. 버튼을 클릭할때 그룹상태를 초기화 하기 위해 아래 코드를 추가합니다.

```
// Reset Groups
$("#btnResetGroups").on("click", function(){
    gridView.groupBy([]);
})
```

1. 버튼을 클릭할때 `직업` 컬럼, `국어` 컬럼, `직업, 국어` 컬럼으로 각각 그룹핑 하도록 아래 코드를 추가 합니다.

```
// 직업컬럼 단일그룹
$("#btnField1Group").on("click", function(){
    gridView.groupBy(["field1"]);
})

// 국어컬럼 단일그룹
$("#btnField4Group").on("click", function(){
    gridView.groupBy(["field4"]);
})

// 직업컬럼, 국어컬럼으로 중첩그룹
$("#btnMultiFieldsGroup").on("click", function(){
    gridView.groupBy(["field1", "field4"]);
})
```

A17 행과 열 고정하기(Fixing)

Jul 27, 2015

RealGridJS RealGrid dataprovider itemmodel fixedcolumn fixedrow fixed fixing 고정

들어가며

이번 강좌에서는 그리드의 행과 열을 고정하는 방법을 배워보겠습니다. 행이나 열이 고정되면 스크롤에서 제외됩니다. 다시 말해서, 고정된 행이나 열은 그리드의 다른 데이터들이 스크롤 될때에도 왼쪽과 위쪽에 고정된채 스크롤되지 않습니다.

이론

행(Row)이나 열(Column)을 고정하기(Fixing)위해서 GridBase.setFixedOptions()함수를 이용합니다. 이 함수의 인자는 FixedOptions라는 클래스이며 반환값은 없습니다. setFixedOptions()함수가 호출되는 즉시 FixedOptions의 속성값에 따라 그리드의 고정상태가 변경됩니다.

행 고정

행을 고정하기 위해서는 `rowCount` 속성에 고정할 행의 갯수를 넘겨 줍니다. 고정된 행은 기본적으로 소팅, 그룹핑, 필터링에서 제외됩니다.

열 고정

열을 고정하기 위해서는 `colCount` 속성에 고정할 열의 갯수를 넘겨 줍니다. 고정된 열은 기본적으로, 사용자가 너비를 조정할 수 없습니다.

초기화

행 고정, 열 고정 값을 초기화 하려면 각각 `rowCount` , `colCount` 의 속성에 `0` 값을 입력 하면 됩니다.

이 외에도 FixedOptions 클래스에는 여러가지 속성이 포함되어 있습니다. FixedOptions에 대한 속성에 대해서는 [B 클래스]강좌에서 좀더 자세히 다루도록 하겠습니다.

실습

1. 실습을 위해 필드와 컬럼의 갯수를 늘렸으며, 데이터도 추가했습니다.
2. 버튼을 클릭할때 행, 열 고정상태를 초기화 하기 위해 아래 코드를 추가합니다.

```
// 행, 열 고정 상태 초기화
$("#btnResetFixing").on("click", function(){
    gridView.setFixedOptions({
        colCount: 0,
        rowCount: 0
    });
})
```

1. 버튼을 클릭할때 각각 행 고정, 열 고정, 행 열 고정하도록 아래 코드를 추가합니다.

```
// 행 고정
$("#btnRowFixing").on("click", function(){
    gridView.setFixedOptions({
        rowCount: 1
    });
})

// 열 고정
$("#btnColumnFixing").on("click", function(){
    gridView.setFixedOptions({
        colCount: 3
    });
})

// 행 열 고정
$("#btnRowColumnFixing").on("click", function(){
    gridView.setFixedOptions({
        colCount: 3,
        rowCount: 1
    });
})
```

A18 컬럼 필터링(Filtering) - I 필터선택상자 사용하기

Jul 29, 2015

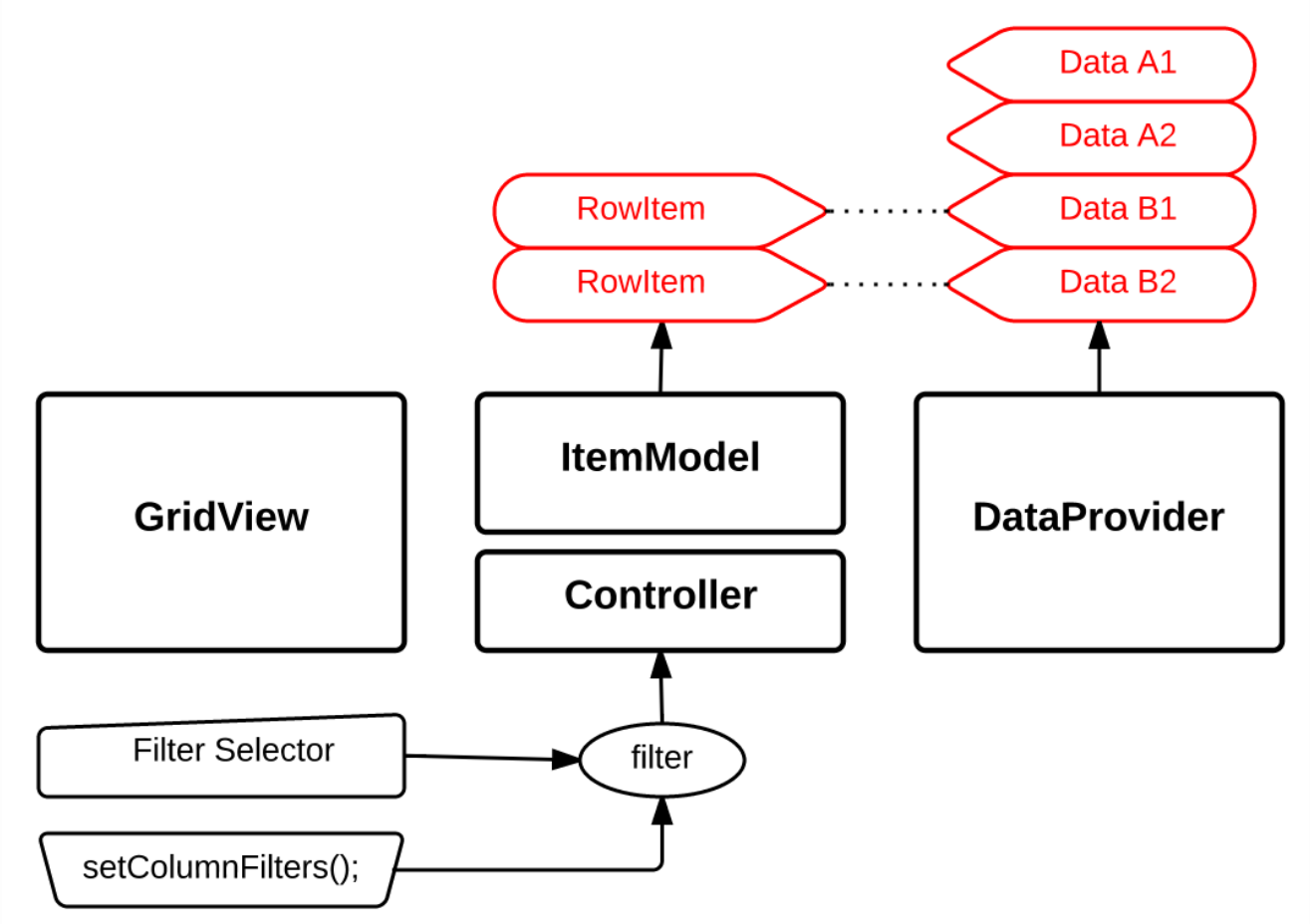
RealGridJS RealGrid 필터 필터링 filter filtering filterselector

들어가며

[A18 강좌]와 [A19 강좌]에서는 필터링에 대해 배워보겠습니다. RealGrid는 다양한 방법으로 필터링이 가능하지만 [A클래스]에서는 기본적인 필터링 방법에 대해서만 다루게 됩니다.

이론

RealGrid에서 필터링(Filtering)이란 특정컬럼의 데이터중 주어진 조건에 해당하는 데이터가 포함된 행만을 그리드에 표시하는 기능을 의미합니다. 즉, 컬럼 단위로 조건을 주어 데이터를 필터링 하는 방식입니다.
아래 그림과 같이 필터선택상자(Filter selector) 또는 필터링 함수(GridBase.setColumnFilters())를 통해 필터링 명령이 전달되면 ItemModel에는 필터링 조건에 맞는 데이터만 남아 있게됩니다.



필터선택상자로 필터링하기

필터선택상자는 컬럼헤더에 표시된 필터 아이콘을 클릭할때 표시되는 선택상자입니다. 필터선택상자에서 선택(체크)된 데이터가 그리드에 표시됩니다. 필터아이콘과 필터선택상자를 표시하려면 해당 컬럼을 생성할때 `filters` 속성을 지정해 주면 됩니다. 아래 실습의 코드를 참조하세요.

setColumnFilters()로 필터링하기

`setColumnFilters(column, filters)` 함수는 필터링 할 데이터 컬럼(DataColumn) 정보와 컬럼필터(ColumnFilter)를 인자로 필터링하는 함수입니다. 자세한 내용은 [A19 강좌]에서 배워보겠습니다.

컬럼의 필터정보 지우기

필터가 지정된 컬럼에서 필터정보를 지우기 위해서는 GridBase.clearColumnFilters()함수를 사용합니다.

실습

1. `직업` 컬럼에 필터를 지정하기 위해 컬럼생성 배열에 필터 정보를 추가합니다. //필드와 연결된 컬럼 배열 객체를 생성합니다.

```
    //필드와 연결된 컬럼 배열 객체를 생성합니다.
    var columns = [
        {
            name: "col1",
            fieldName: "field1",
            header : {
                text: "직업"
            },
            width : 60,
            filters : [{
                name: "가수",
                criteria: "value = '가수'"
            }, {
                name: "배우",
                criteria: "value = '배우'"
            }]
        },
        {
        ...
```

1. 버튼을 클릭할때 `직업` 컬럼에 지정된 필터정보를 제거하기 위해 clearColumnFilters()함수는 호출합니다.

```
    // 필터링된 컬럼의 필터정보 지우기
    $("#btnClearColumnFilters").on("click", function(){
        gridView.clearColumnFilters("col1");
    })
```

A19 컬럼 필터링(Filtering) - II setColumnFilters()함수 사용하기

Jul 30, 2015

RealGridJS RealGrid 필터 필터링 filter filtering setColumnFilters

들어가며

이번 강좌에서는 지난[A18 강좌]에 이어 GridBase.setColumnFilters()함수를 이용해 필터링하는 방법에 대해 배워보겠습니다.

이론

기본적인 필터링 함수인 GridBase.setColumnFilters()는 데이터컬럼(DataColumn)과 컬럼필터(ColumnFilter)를 인자로 합니다. 첫 번째 인자인 데이터컬럼은 필터링할 컬럼을 지시합니다. 두 번째 인자인 컬럼필터는 필터선택상자에 표시될 필터정보를 지시합니다.

setColumnFilters()함수의 첫 번째 인자인 데이터컬럼에는 DataColumn객체를 넘겨도 되지만 단순히 컬럼이름(ColumnName)만 string형으로 넘겨주어도 됩니다. 아래 코드는 데이터컬럼에 컬럼이름만 넘기는 경우와 데이터컬럼 객체를 넘기는 두 가지 경우의 코드를 보여주고 있습니다. 두 경우 모두 동일한 동작을 하게 됩니다.

```
//컬럼 이름만 넘기는 경우
gridView.setColumnFilters("col1", filters)

//데이터컬럼 객체를 넘기는 경우
var dataColumn = {"name" : "col1"}
gridView.setColumnFilters(dataColumn, filters);
```

컬럼필터(ColumnFilter)객체에는 `name`, `criteria`, `text`, `description`, `active`라는 속성이 있지만 이번 강좌에서는 `name`, `criteria`, `text`속성만 알아보겠습니다.

- `name`속성은 필터를 구분하기 위한 이름입니다.
- `criteria`속성은 필터의 조건을 입력하기 위한 Expression속성 입니다.
- `text`속성은 필터선택창에 표시될 문자열입니다. 입력하지 않으면 `name`속성이 표시됩니다.

실습

1. 버튼을 클릭하면 `직업`컬럼에 필터를 지정하기 위해 filters라는 ColumnFilter객체를 만들고 필터 정보를 입력합니다.
2. setColumnFilters()함수로 직업컬럼에 필터를 적용하기 위해 "col1"이라는 직업컬럼 필터의 이름을 첫 번째 인자로 넘겨줍니다.
3. ColumnFilter객체인 filters를 두 번째 인자로 넘겨줍니다.

```
// setColumnFilters()함수로 직업컬럼에 필터 적용
$("#btnSetColumnFilters").on("click", function(){
    var filters = [{
            name: "가수",
            criteria: "value = '가수'",
            text: "가수만 걸러주세요."
        }, {
            name: "배우",
            criteria: "value = '배우'"
        }];

    gridView.setColumnFilters("col1", filters);
})
```

1. 버튼을 클릭할때 `직업` 컬럼에 지정된 필터정보를 제거하기 위해 clearColumnFilters()함수는 호출합니다.

```
// 필터링된 컬럼의 필터정보 지우기
$("#btnClearColumnFilters").on("click", function(){
    gridView.clearColumnFilters("col1");
})
```

A20 셀 선택하기(Selecting) - SelectOptions

Jul 31, 2015

RealGridJS RealGrid 선택 selection selecting 선택블럭 선택영역 selectoptions selectstyle setselectoptions

들어가며

[A20 강좌], [A21 강좌], [A22 강좌]에서는 RealGrid의 셀 선택 기능에 대해 배워보겠습니다.

이론

셀 선택 기능이란 사용자가 마우스 또는 트랙패드를 이용하여 그리드의 특정 셀을 선택하거나 셀의 범위를 선택하는 기능을 말합니다.

셀 선택에는 6가지 SelectionStyle이 있으며 이 클래스는 SelectOptions에 style속성의 값으로 설정할 수 있습니다.

각각의 선택 스타일을 간단한 설명과 함께 캡쳐된 화면을 나열해 보겠습니다.

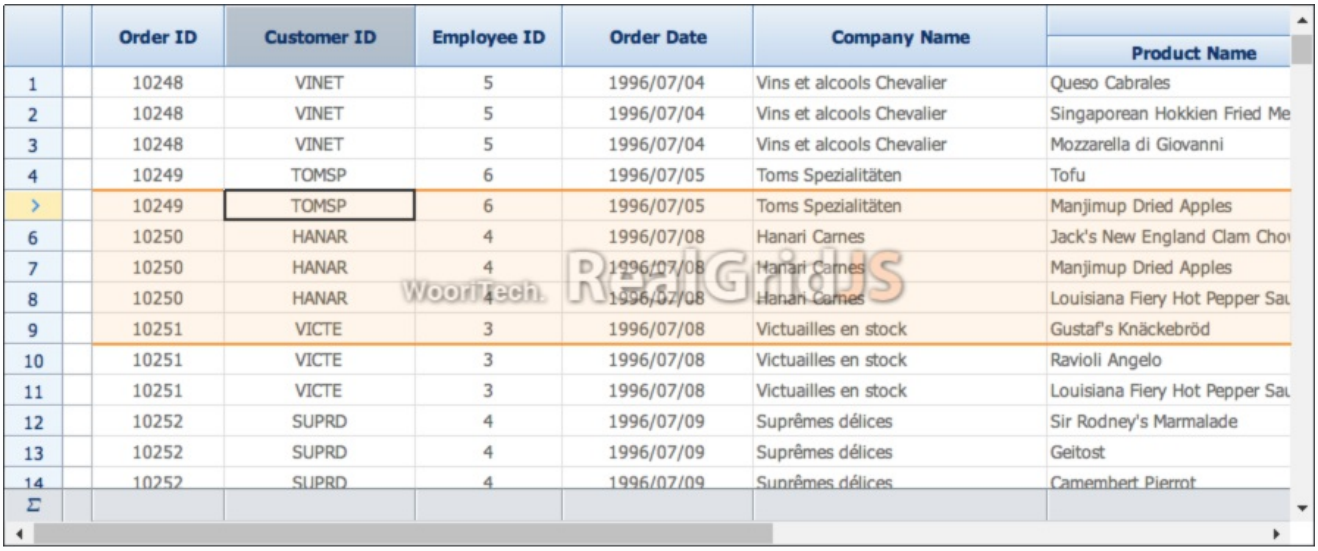
1. BLOCK

하나의 셀 또는 블럭 형태로 셀 범위를 선택할 수 있습니다. 이 설정은 그리드 생성시 기본값이므로 따로 설정할 필요가 없습니다.

	Order ID	Customer ID	Employee ID	Order Date	Company Name	Product Name
1	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Queso Cabrales
2	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Singaporean Hokkien Fried Me
3	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Mozzarella di Giovanni
4	10249	TOMSP	6	1996/07/05	Toms Spezialitäten	Tofu
>	10249	TOMSP	6	1996/07/05	Toms Spezialitäten	Manjimup Dried Apples
6	10250	HANAR	4	1996/07/08	Hanari Carnes	Jack's New England Clam Cho
7	10250	HANAR	4	1996/07/08	Hanari Carnes	Manjimup Dried Apples
8	10250	HANAR	4	1996/07/08	Hanari Carnes	Louisiana Fiery Hot Pepper Sau
9	10251	VICTE	3	1996/07/08	Victuailles en stock	Gustaf's Knäckebröd
10	10251	VICTE	3	1996/07/08	Victuailles en stock	Ravioli Angelo
11	10251	VICTE	3	1996/07/08	Victuailles en stock	Louisiana Fiery Hot Pepper Sau
12	10252	SUPRD	4	1996/07/09	Suprêmes délices	Sir Rodney's Marmalade
13	10252	SUPRD	4	1996/07/09	Suprêmes délices	Geitost
14	10252	SUPRD	4	1996/07/09	Suprêmes délices	Camembert Pierrot
Σ						

2. ROWS

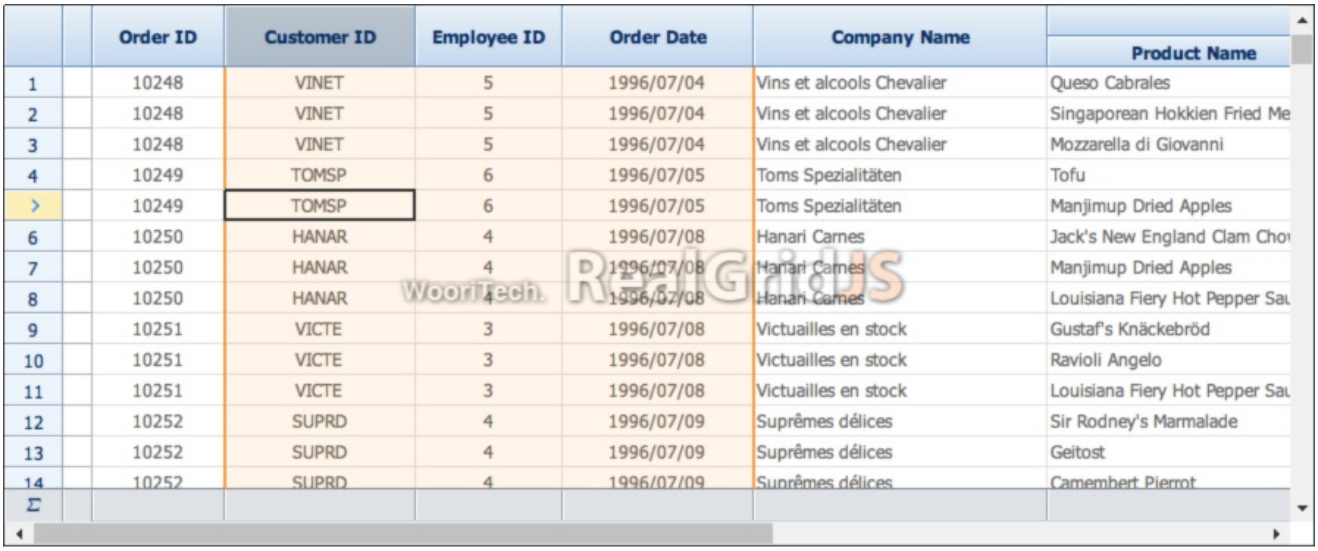
한 줄의 행 또는 여러줄의 행을 셀 범위로 선택할 수 있습니다.



	Order ID	Customer ID	Employee ID	Order Date	Company Name	Product Name
1	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Queso Cabrales
2	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Singaporean Hokkien Fried Me
3	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Mozzarella di Giovanni
4	10249	TOMSP	6	1996/07/05	Toms Spezialitäten	Tofu
>	10249	TOMSP	6	1996/07/05	Toms Spezialitäten	Manjimup Dried Apples
6	10250	HANAR	4	1996/07/08	Hanari Carnes	Jack's New England Clam Cho
7	10250	HANAR	4	1996/07/08	Hanari Carnes	Manjimup Dried Apples
8	10250	HANAR	4	1996/07/08	Hanari Carnes	Louisiana Fiery Hot Pepper Sau
9	10251	VICTE	3	1996/07/08	Victuailles en stock	Gustaf's Knäckebröd
10	10251	VICTE	3	1996/07/08	Victuailles en stock	Ravioli Angelo
11	10251	VICTE	3	1996/07/08	Victuailles en stock	Louisiana Fiery Hot Pepper Sau
12	10252	SUPRD	4	1996/07/09	Suprêmes délices	Sir Rodney's Marmalade
13	10252	SUPRD	4	1996/07/09	Suprêmes délices	Geitost
14	10252	SUPRD	4	1996/07/09	Suprêmes délices	Camembert Pierrot
Σ						

3. COLUMNS

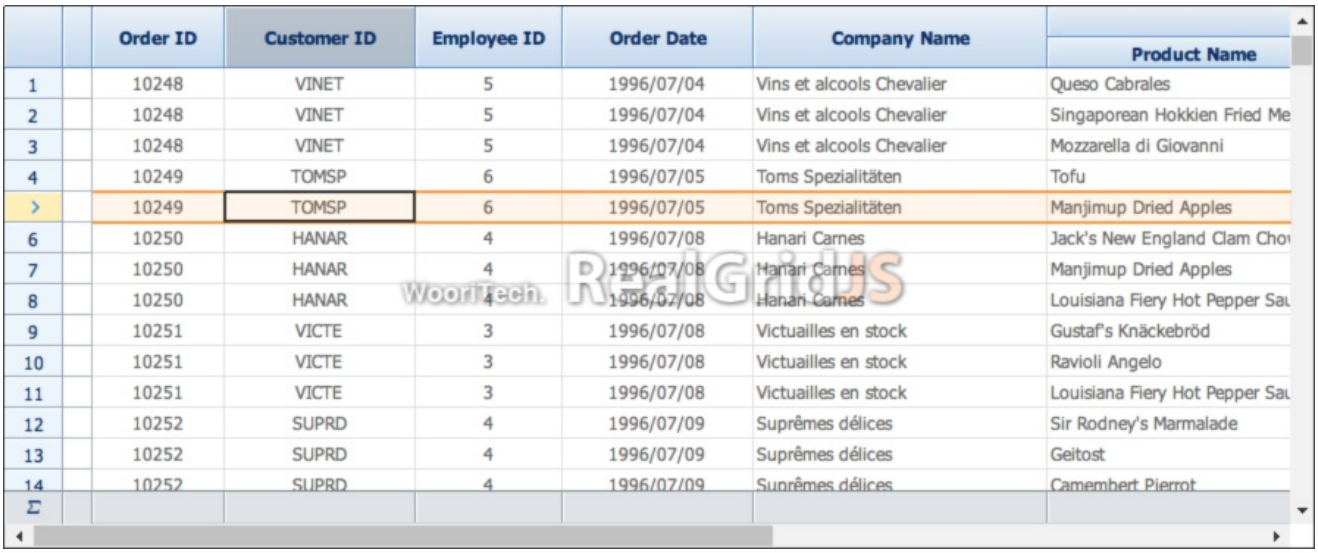
하나의 컬럼 또는 여러개의 컬럼을 셀 범위로 선택할 수 있습니다.



	Order ID	Customer ID	Employee ID	Order Date	Company Name	Product Name
1	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Queso Cabrales
2	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Singaporean Hokkien Fried Me
3	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Mozzarella di Giovanni
4	10249	TOMSP	6	1996/07/05	Toms Spezialitäten	Tofu
>	10249	TOMSP	6	1996/07/05	Toms Spezialitäten	Manjimup Dried Apples
6	10250	HANAR	4	1996/07/08	Hanari Carnes	Jack's New England Clam Cho
7	10250	HANAR	4	1996/07/08	Hanari Carnes	Manjimup Dried Apples
8	10250	HANAR	4	1996/07/08	Hanari Carnes	Louisiana Fiery Hot Pepper Sau
9	10251	VICTE	3	1996/07/08	Victuailles en stock	Gustaf's Knäckebröd
10	10251	VICTE	3	1996/07/08	Victuailles en stock	Ravioli Angelo
11	10251	VICTE	3	1996/07/08	Victuailles en stock	Louisiana Fiery Hot Pepper Sau
12	10252	SUPRD	4	1996/07/09	Suprêmes délices	Sir Rodney's Marmalade
13	10252	SUPRD	4	1996/07/09	Suprêmes délices	Geitost
14	10252	SUPRD	4	1996/07/09	Suprêmes délices	Camembert Pierrot
Σ						

4. SINGLE_ROW

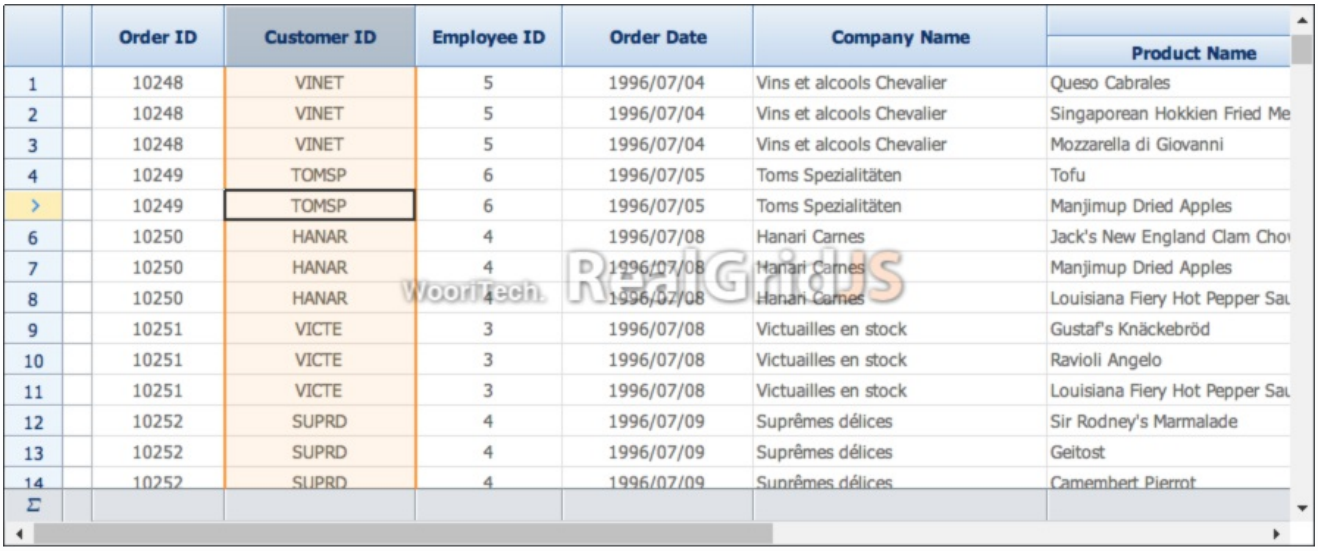
한 줄의 행만을 선택할 수 있습니다.



	Order ID	Customer ID	Employee ID	Order Date	Company Name	Product Name
1	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Queso Cabrales
2	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Singaporean Hokkien Fried Me
3	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Mozzarella di Giovanni
4	10249	TOMSP	6	1996/07/05	Toms Spezialitäten	Tofu
>	10249	TOMSP	6	1996/07/05	Toms Spezialitäten	Manjimup Dried Apples
6	10250	HANAR	4	1996/07/08	Hanari Carnes	Jack's New England Clam Cho
7	10250	HANAR	4	1996/07/08	Hanari Carnes	Manjimup Dried Apples
8	10250	HANAR	4	1996/07/08	Hanari Carnes	Louisiana Fiery Hot Pepper Sau
9	10251	VICTE	3	1996/07/08	Victuailles en stock	Gustaf's Knäckebröd
10	10251	VICTE	3	1996/07/08	Victuailles en stock	Ravioli Angelo
11	10251	VICTE	3	1996/07/08	Victuailles en stock	Louisiana Fiery Hot Pepper Sau
12	10252	SUPRD	4	1996/07/09	Suprêmes délices	Sir Rodney's Marmalade
13	10252	SUPRD	4	1996/07/09	Suprêmes délices	Geitost
14	10252	SUPRD	4	1996/07/09	Suprêmes délices	Camembert Pierrot
Σ						

5. SINGLE_COLUMN

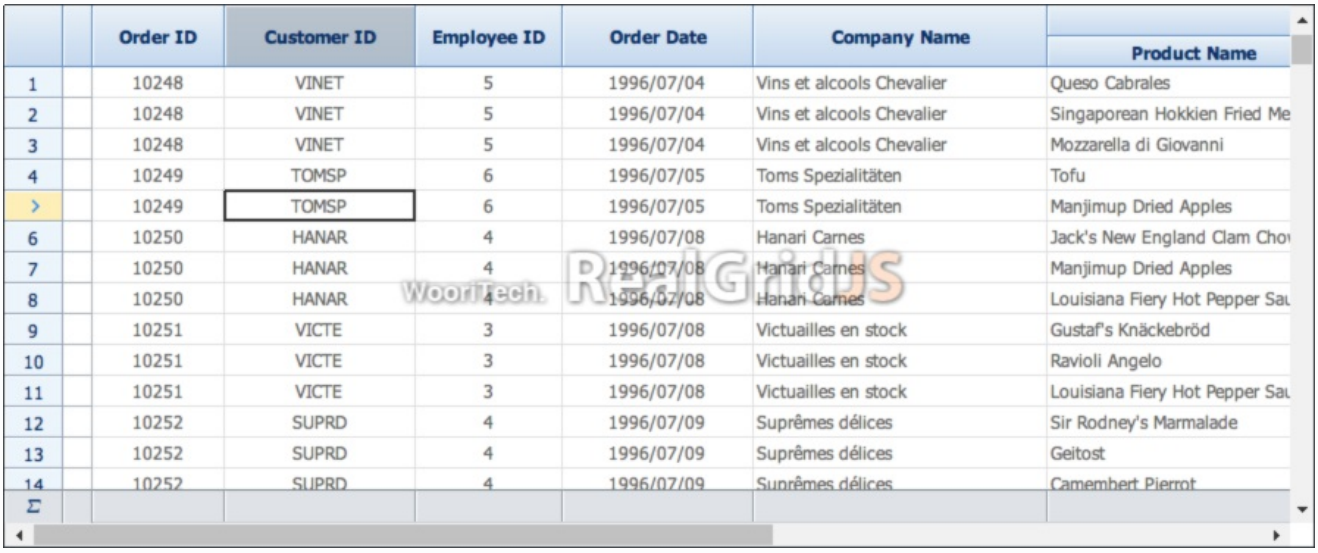
하나의 컬럼만을 선택할 수 있습니다.



	Order ID	Customer ID	Employee ID	Order Date	Company Name	Product Name
1	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Queso Cabrales
2	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Singaporean Hokkien Fried Me
3	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Mozzarella di Giovanni
4	10249	TOMSP	6	1996/07/05	Toms Spezialitäten	Tofu
5	10249	TOMSP	6	1996/07/05	Toms Spezialitäten	Manjimup Dried Apples
6	10250	HANAR	4	1996/07/08	Hanari Carnes	Jack's New England Clam Cho
7	10250	HANAR	4	1996/07/08	Hanari Carnes	Manjimup Dried Apples
8	10250	HANAR	4	1996/07/08	Hanari Carnes	Louisiana Fiery Hot Pepper Sau
9	10251	VICTE	3	1996/07/08	Victuailles en stock	Gustaf's Knäckebröd
10	10251	VICTE	3	1996/07/08	Victuailles en stock	Ravioli Angelo
11	10251	VICTE	3	1996/07/08	Victuailles en stock	Louisiana Fiery Hot Pepper Sau
12	10252	SUPRD	4	1996/07/09	Suprêmes délices	Sir Rodney's Marmalade
13	10252	SUPRD	4	1996/07/09	Suprêmes délices	Geitost
14	10252	SUPRD	4	1996/07/09	Suprêmes délices	Camembert Pierrot
Σ						

6. NONE

셀을 선택 할 수 없습니다.



	Order ID	Customer ID	Employee ID	Order Date	Company Name	Product Name
1	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Queso Cabrales
2	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Singaporean Hokkien Fried Me
3	10248	VINET	5	1996/07/04	Vins et alcools Chevalier	Mozzarella di Giovanni
4	10249	TOMSP	6	1996/07/05	Toms Spezialitäten	Tofu
5	10249	TOMSP	6	1996/07/05	Toms Spezialitäten	Manjimup Dried Apples
6	10250	HANAR	4	1996/07/08	Hanari Carnes	Jack's New England Clam Cho
7	10250	HANAR	4	1996/07/08	Hanari Carnes	Manjimup Dried Apples
8	10250	HANAR	4	1996/07/08	Hanari Carnes	Louisiana Fiery Hot Pepper Sau
9	10251	VICTE	3	1996/07/08	Victuailles en stock	Gustaf's Knäckebröd
10	10251	VICTE	3	1996/07/08	Victuailles en stock	Ravioli Angelo
11	10251	VICTE	3	1996/07/08	Victuailles en stock	Louisiana Fiery Hot Pepper Sau
12	10252	SUPRD	4	1996/07/09	Suprêmes délices	Sir Rodney's Marmalade
13	10252	SUPRD	4	1996/07/09	Suprêmes délices	Geitost
14	10252	SUPRD	4	1996/07/09	Suprêmes délices	Camembert Pierrot
Σ						

SelectionStyle을 설정하려면 GridBase.setSelectOptions(options)함수를 사용합니다. 함수의 인자로 SelectOptions객체를 넘겨줍니다.

실습

1. 버튼을 클릭하면 SelectionStyle을 설정하기 위해 setSelectOptions()함수를 호출 합니다. SelectionStyle종류에 따라 각각의 버튼 이벤트 함수를 작성합니다.

```
// SelectionStyle을 BLOCK으로 지정하는 버튼 클릭
$("#btnSetSelectOptionsBlock").on("click", function(){
    gridView.setSelectOptions({"style" : "block"});
})

// SelectionStyle을 ROWS로 지정하는 버튼 클릭
$("#btnSetSelectOptionsRows").on("click", function(){
    gridView.setSelectOptions({"style" : "rows"});
})

// SelectionStyle을 COLUMNS로 지정하는 버튼 클릭
$("#btnSetSelectOptionsColumns").on("click", function(){
    gridView.setSelectOptions({"style" : "columns"});
})

// SelectionStyle을 SINGLE_ROW로 지정하는 버튼 클릭
$("#btnSetSelectOptionsSingleRow").on("click", function(){
    //SeleectOptions객체를 만들어 인자로 넘겨도 결과는 같습니다.
    var selectOptions = {"style" : "singleRow"};
    gridView.setSelectOptions(selectOptions);
})

// SelectionStyle을 SINGLE_COLUMN으로 지정하는 버튼 클릭
$("#btnSetSelectOptionsSingleColumn").on("click", function(){
    gridView.setSelectOptions({"style" : "singleColumn"});
})

// SelectionStyle을 NONE으로 지정하는 버튼 클릭
$("#btnSetSelectOptionsNone").on("click", function(){
    gridView.setSelectOptions({"style" : "none"});
})
```

A21 셀 선택하기(Selecting) - 선택한 셀의 값 가져오기

Aug 5, 2015

RealGridJS RealGrid 선택 selection selecting 선택블럭 선택영역 selectoptions selectstyle setselectoptions getSelectionData

들어가며

[A20 강좌]에서는 RealGrid에서 셀을 선택하는 방법에 대해 알아보았습니다. 이번 강좌에서는 선택한 범위에 해당하는 셀의 값을 가져오는 방법에 대해 배워보겠습니다.

이론

선택된 범위에 대한 셀의 값을 가져오려면 GridBase.getSelectionData()함수를 사용하면 됩니다. 이 함수는 아래와 같이 정의되어 있으며 `maxRows`인자는 함수에서 반환할 행의 수를 입력합니다.

```
getSelectionData(maxRows);
```

실습

1. 드롭다운 메뉴를 선택하면 SelectionOptions의 SelectionStyle을 변경하도록 작성합니다. Bootstrap의 dropdown-menu는 <a>태그를 사용하기 때문에 버튼과 마찬가지로 click()이벤트에서 처리하도록 작성합니다.

```
// SelectionStyle을 BLOCK으로 지정하는 버튼 클릭
$("#btnSetSelectOptionsBlock").on("click", function(){
    gridView.setSelectOptions({"style" : "block"});
})

// SelectionStyle을 ROWS로 지정하는 버튼 클릭
$("#btnSetSelectOptionsRows").on("click", function(){
    gridView.setSelectOptions({"style" : "rows"});
})

// SelectionStyle을 COLUMNS로 지정하는 버튼 클릭
$("#btnSetSelectOptionsColumns").on("click", function(){
    gridView.setSelectOptions({"style" : "columns"});
})

// SelectionStyle을 SINGLE_ROW로 지정하는 버튼 클릭
$("#btnSetSelectOptionsSingleRow").on("click", function(){
    //SeleectOptions객체를 만들어 인자로 넘겨도 결과는 같습니다.
    var selectOptions = {"style" : "singleRow"};
    gridView.setSelectOptions(selectOptions);
})

// SelectionStyle을 SINGLE_COLUMN으로 지정하는 버튼 클릭
$("#btnSetSelectOptionsSingleColumn").on("click", function(){
    gridView.setSelectOptions({"style" : "singleColumn"});
})

// SelectionStyle을 NONE으로 지정하는 버튼 클릭
$("#btnSetSelectOptionsNone").on("click", function(){
    gridView.setSelectOptions({"style" : "none"});
})
```

2. 두 개의 버튼을 클릭 할때 getSelectionData()함수를 사용하여 각각 선택된 범위의 데이터를 가져오는 코드와 선택된 범위중 2개행의 데이터만 거져오는 코드를 작성합니다.

```
// 선택된 범위의 모든 행에 대한 데이터를 가져옵니다.
$("#btnGetSelectionData").on("click", function(){
    var selData = gridView.getSelectionData();
    selData = JSON.stringify(selData);
    $("#allRowSelectionData").text(selData);
})

// 선택된 범위중 2개행에 대한 데이터만 가져옵니다.
$("#btnGet2RowSelectionData").on("click", function(){
    var selData = gridView.getSelectionData(2);
    selData = JSON.stringify(selData);
    $("#twoRowSelectionData").text(selData);
})
```

A22 셀 선택하기(Selecting) - 동적으로 셀 선택하기

Aug 7, 2015

RealGridJS RealGrid 선택 selection selecting 선택블럭 선택영역 selectionitem selectstyle setSelection getSelection clearSelection

들어가며

[A20 강좌]와 [A21 강좌]에서는 사용자가 마우스를 이용해서 선택영역을 지정하는 방법 및 스타일과 선택영역의 데이터를 가져오는 방법에 대해 알아보았습니다. 이번 강좌에서는 동적(프로그램적)으로 그리드의 특정영역을 선택하는 방법에 대해 배워보겠습니다.

이론

RealGrid에는 그리드의 선택영역 정보를 가진 SelectionItem이란 클래스가 있습니다. SelectionItem은 선택영역의 시작점과 끝점의 정보를 포함하고 있으며 선택의 종류를 구분하기 위한 SelectionStyle정보도 포함합니다. 선택영역 정보(SelectionItem)을 그리드에 적용하거나 가져오기 위해 다음과 같은 함수가 준비되어 있습니다.

1. 선택영역을 가져오는 함수: GridBase.getSelection();
2. 선택영역을 적용하는 함수: GridBase.setSelection();
3. 선택영역을 지우는 함수: GridBase.clearSelection();

SelectionItem에는 영역을 선택하는 방법에 따라 `style`속성을 지정해 주면 됩니다. `style`속성에는 “bolck”, “rows”, “columns”의 값을 입력하고 각각의 스타일에 따라 다음과 같은 범위지정 속성에 값을 입력합니다.

- style: “block”인 경우 startItem, endItem 또는 startRow, endRow 와 startColumn, endColumn
- style: “rows”인 경우 startItem, endItem 또는 startRow, endRow
- style: “columns”인 경우 startColumn, endColumn

실습

1. 버튼을 클릭하면 Selection style에 따라 블럭, 행, 컬럼을 선택하도록 setSelection()함수를 호출하는 코드를 작성합니다.

```
// 블럭단위 선택영역 지정하기
$("#btnSetBlockSelection").on("click", function(){
    var selection = {
        style: "block",
        startItem: 2,
        endItem: 4,
        startColumn: "col3",
        endColumn: "col5"
    }
    gridView.setSelection(selection);
})

// 행단위 선택영역 지정하기
$("#btnSetRowSelection").on("click", function(){
    var selection = {
        style: "rows",
        startRow: 3,
        endRow: 5
    }
    gridView.setSelection(selection);
})

// 컬럼단위 선택영역 지정하기
$("#btnSetColumnSelection").on("click", function(){
    var selection = {
        style: "columns",
        startColumn: "col4",
        endColumn: "col6"
    }
    gridView.setSelection(selection);
})
```

2. 버튼을 클릭하면 선택영역정보(SelectionItem)를 가져오기위해 getSelection()함수를 호출하는 코드를 작성합니다.

```
// SelectionItem 가져오기
$("#btnGetSelection").on("click", function(){
    var selection = gridView.getSelection();
    alert(JSON.stringify(selection));
})
```

3. 버튼을 클릭하면 선택영역을 제거하기위해 clearSelection()함수를 호출하는 코드를 작성합니다.

```
// 선택영역 모두 지우기
$("#btnClearSelection").on("click", function(){
    gridView.clearSelection();
})
```

A23 데이터 편집하기(Editing)

Aug 10, 2015

RealGridJS RealGrid 편집 RowState created updated deleted commit createanddeleted

들어가며

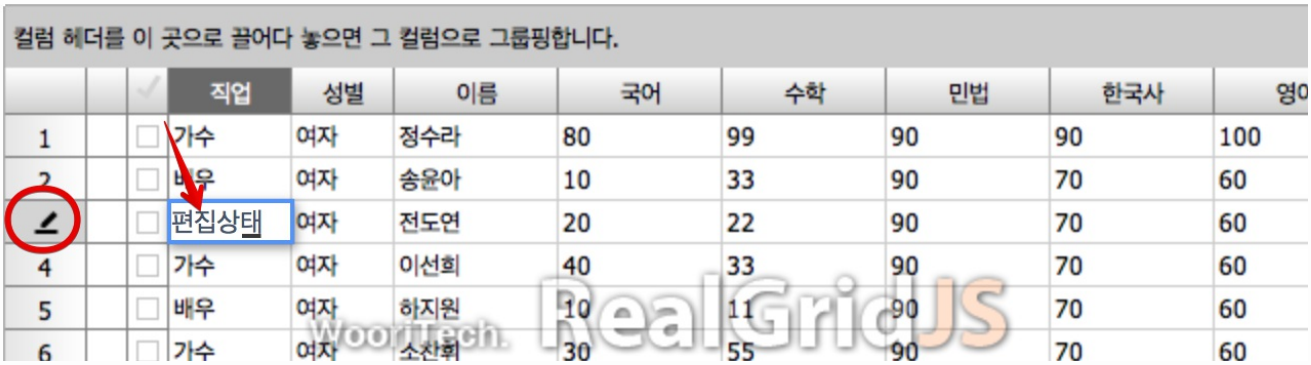
이번 강좌에서는 RealGrid의 강력한 편집기능에 대해 알아보겠습니다. 편집기능을 하나의 강좌에서 다루기에는 너무 많은 내용이지만 요약하여 기본적인 내용만 배워보도록 하겠습니다.

이론

RealGrid에서 편집은 Data와 Item으로 구분해서 생각해볼 필요가 있습니다. Data와 Item에 대한 기본적인 이해는 [A11 강좌]에서 배울 수 있습니다.

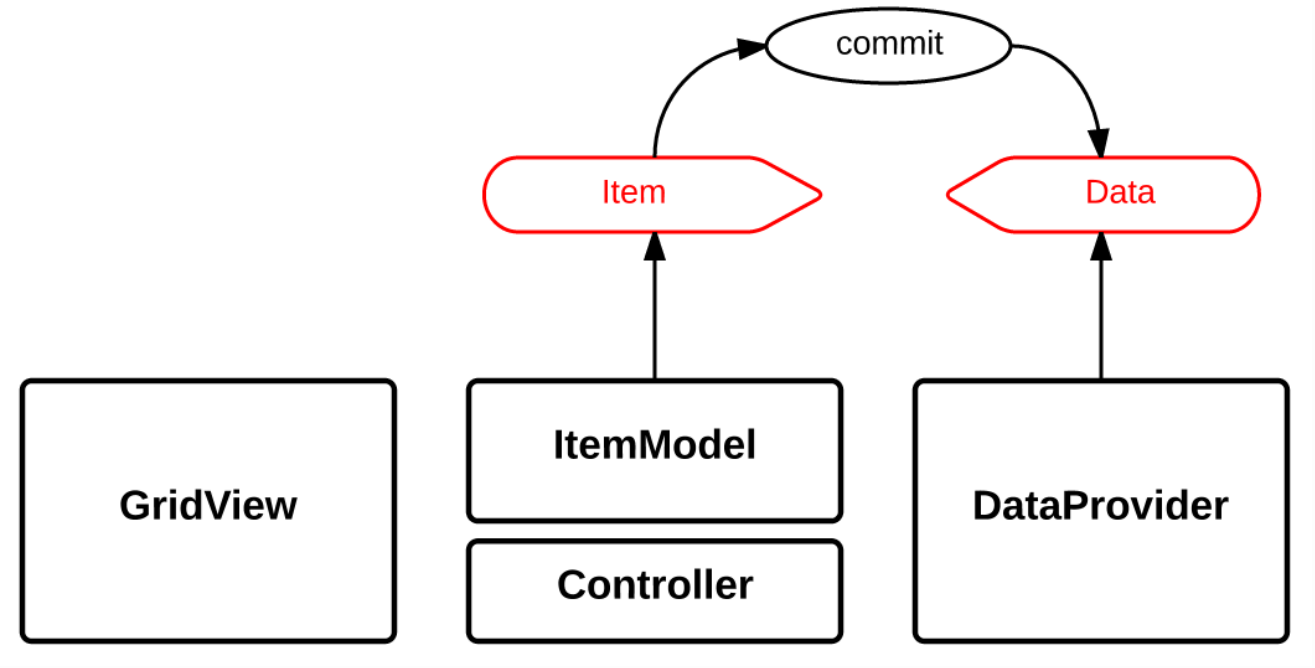
편집상태

RealGrid는 그리드에서 직접 키보드등 입력장치를 통해 데이터를 편집 할 수 있습니다. 데이터가 표시된 그리드의 셀(Cell)위에서 문자나 숫자 키를 입력하면 그리드는 즉시 편집모드로 들어갑니다. 이때 해당 행의 상태는 편집상태(Editing)가 되며 인디케이터(Indicator)에는 편집상태를 알리는 아이콘이 표시됩니다. 일반적으로 편집상태는 아래 화면과 같습니다.



Commit

행이 편집상태일때 셀(Cell)의 값은 Data의 값이 아닌 Item의 값입니다. Item의 값은 Data에 즉시 전달되지 않고 Commit이라는 명령에 의해 DataProvider에 반영하게 됩니다. 아래 그림은 변경된 Item의 값이 Commit명령에 의해 Data에 반영되는 과정을 표현한 것입니다.
Commit명령은 GridBase.Commit()함수로 실행되거나 행이 변경될때 자동으로 실행됩니다.



행의 상태(RowState)

그리드에서 편집에 해당하는 기능은 일반적으로 추가, 수정, 삭제가 있습니다. RealGrid는 기능에 따라 행의 상태(RowState)가 달라지고 행의 상태는 상태바(StateBar)에 아이콘으로 표시됩니다.
아래 표에는 기능별 상태와 상태바의 아이콘이 어떻게 달라지는지 그 차이를 나타내고 있습니다.

기능	키보드 동작	API	상태(RowState)	상태바 아이콘
조회			none	
행 삽입(Insert)	`INSERT`	beginInsertRow()	created	
행 추가 (Append)	마지막행에서 `↓`	beginAppendRow()	created	
행 수정(Update)	`F2` 또는 값변경	beginUpdateRow()	updated	
행 삭제(Delete)	`CTRL` + `DELETE`	deleteSelection()	deleted	
행 삽입 또는 추가 후 삭제	`INSERT` 후 `CTRL` + `DELETE`		createAndDeleted	

- 위 API는 모두 GridView클래스의 메서드 입니다.
- 상태바(StatusBar)가 어디를 나타내는지는 데모사이트 Component페이지에서 확인하세요.
- 행 삽입(Insert)은 현재 선택된 행의 바로 위에 새로운 행이 추가됩니다.
- 행 추가(Append)는 맨 마지막행의 바로 아래에 새로운 행이 추가됩니다.

실습을 통해 각 기능의 작동 방법과 기능에 따라 상태가 어떻게 달라지는지 확인해보겠습니다.

실습

1. 행의 추가, 삽입, 삭제가 가능하도록 EditOptions의 insertable속성과 appendable, deletable속성을 true로 수정합니다.

```
//행 삽입과 행 추가, 행 삭제가 가능하도록 옵션 조정
gridView.setEditOptions({
    insertable: true,
    appendable: true,
    deletable: true
})
```

2. 버튼 클릭시 행 삽입, 행 추가, 행 삭제 기능을 수행하는 코드작성

```
// 현재행 위해 새로운 행 추가위해 beginInserRow()함수 실행
$("#btnInsert").click( function() {
    var current = gridView.getCurrent();
    gridView.beginInsertRow(current.itemIndex);
})

// 맨 마지막 행 추가위해 beginAppendRow()함수 실행
$("#btnAppend").click(function() {
    gridView.beginAppendRow();
})

// 선택된 행 삭제를 위해 deleteSelection()함수 실행
$("#btnDelete").click(function() {
    gridView.deleteSelection();
})
```

3. 버튼을 클릭하면 현재 그리드의 상태별 RowId정보를 가져와서 alert하는 코드 작성

```
$("#btnGetStateRows").click(function() {
    var rowStates = {
        "created": dataProvider.getStateRows("created"),
        "updated": dataProvider.getStateRows("updated"),
        "deleted": dataProvider.getStateRows("deleted"),
        "createAndDeleted": dataProvider.getStateRows("createAndDeleted"),
        "none": dataProvider.getStateRows("none")
    };

    alert(JSON.stringify(rowStates));
})
```

A24 데이터 불러오기(Data Loading)

Aug 11, 2015

RealGridJS RealGrid 편집 RowState created updated deleted commit createanddeleted

들어가며

이번 강좌에서는 DataProvider를 통해 RealGridJS에 데이터를 읽어오는 방법에 대해 배워보겠습니다.

이론

읽어올 데이터의 원본은 원격에 존재할 수도 있고 로컬에 존재할 수도 있습니다. 이번 강좌에서는 원격에 존재하는 데이터를 읽어오기 위해 jQuery의 ajax()를 사용합니다. 로컬에 존재하는 데이터도 원본 데이터셋의 유형과 방법은 본 강좌의 내용과 동일하기 때문에 별도로 설명하지는 않겠습니다.

RealGrid가 읽어올 수 있는 원본 데이터의 유형은 JSON, XML, CSV 세가지 입니다. 세가지 유형의 데이터를 읽어오기 위해 각각 LocalDataProvider 함수를 사용할수 있으며 그 종류는 다음과 같습니다.

- JSON : LocalDataProvider.fillJsonData()
- XML : LocalDataProvider.fillXmlData()
- CSV : LocalDataProvider.fillCsvData()

세 함수는 모두 두 개의 인자를 가지고 있습니다. 첫번째 인자는 data이고 두번째 인자는 DataFillOptions라는 옵션입니다. 이 옵션에는 fillMode라는 이름의 속성이 있는데 이 속성은 DataFillMode라는 옵션입니다. DataFillOption은 데이터의 갯수 데이터를 채우기 시작할 행의 itemIndex등의 정보를 담고 있으며, DataFillMode는 행을 추가(Append)할 것인지 삽입(Insert)할 것인지 아니면 수정(Update)할 것인지 등의 정보를 결정합니다.

첫번째 인자인 data는 유형에 따라 일반적으로 다음과 같은 형식을 유지해야 합니다. 단, 형식이 다른 경우 옵션으로 조정 할 수 있는 방법도 있습니다. 보다 자세한 내용은 B-Class에서 다룰 예정입니다.

JSON Data Format

```
[
    ["가수", "여자", "이선희", "40", "33", "90", "70", "60", "100", "80"],
    ["배우", "여자", "하지원", "10", "11", "90", "70", "60", "100", "80"],
    ["가수", "여자", "소찬휘", "30", "55", "90", "70", "60", "100", "80"],
    ["가수", "여자", "박정현", "40", "22", "90", "70", "60", "100", "80"],
    ["배우", "여자", "전지현", "20", "44", "90", "70", "60", "100", "80"]
]
```

XML Data Format

```
<rows>
    <row field1="배우" field2="여자" field3="전도연" field4="20" field5="22" field6="90" fi
</rows>
```

CSV Data Format

```
가수, 여자, 정수라, 80, 99, 90, 90, 100, 100, 90
배우, 여자, 송윤아, 10, 33, 90, 70, 60, 100, 80
배우, 여자, 송윤아, 10, 33, 90, 70, 60, 100, 80
```

실습에서는 유형에 따른 원격데이터를 그리드에 입력하는 방법을 직접 구현해 보겠습니다.

실습

1. JSON, XML, CSV 각 유형의 버튼을 클릭하면 각 유형에 맞는 fillData함수를 호출하여 그리드에 데이터를 채우는 코드를 작성 합니다. 이때, 원격의 데이터를 가져오기 위해 jQuery의 ajax()함수를 써서 GET Method를 이용해 데이터를 가져옵니다.

```
//현재그리드의 데이터를 모두 지우고 8행의 CSV유형 데이터를 가져와서 채웁니다.
$("#btnFillCSVData").click(function() {
     $.ajax({
        url: "/data/samples/sampleCSVData.csv",
        cache: false
    })
    .done(function( data ) {
        dataProvider.fillCsvData(data, { count: 8, fillMode: "set" });
    });
})

//현재 그리드의 데이터 맨 마지막행 뒤에 5행의 JSON유형 데이터를 가져와서 채웁니다.
$("#btnFillJSONData").click(function() {
     $.ajax({
        url: "/data/samples/sampleJSONData.json",
        cache: false
    })
    .done(function( data ) {
        dataProvider.fillJsonData(data, { fillMode: "append" });
    });
})

//현재 그리드의 데이터중 3행의 데이터를 가져온 XML 데이터로 덮어쓰기 합니다.
$("#btnFillXMLData").click(function() {
     $.ajax({
        url: "/data/samples/sampleXMLData.xml",
        cache: false
    })
    .done(function( data ) {
        dataProvider.fillXmlData(data, { count: 1, fillPos: 2, fillMode: "update"
    });
})
```

A25 에디터(Editor)와 셀의 값

Aug 12, 2015

RealGridJS RealGrid 편집기 editor datatype 자료형 subdatatype datetime number date dropdown combo

들어가며

이번 강좌에서는 RealGrid의 자료형(DataType)과 편집기(Editor)의 종류에 대해 배워보겠습니다. 추가로 그리드의 데이터를 가져오거나 넣는 방법에 대해 기본적인 내용만 알아보겠습니다.

이론

자료형(DataType)

자료형(DataType)은 DataProvider에서 데이터를 구성하는 DataField의 유형을 지정하며 아래와 같은 종류가 있습니다.

- TEXT : 문자형
- BOOL : 논리형
- NUMBER : 숫자형
- DATETIME : 날짜형

기본자료형을 제한하기 위한 용도로 하위자료형(SubDataType)이 있으며 아래와 같은 종류가 있습니다.

- CHAR
- UNUM
- INT
- UINT
- DATE

편집기(Editor)

편집기(Editor)는 사용자가 셀의 값을 수정할 수 있도록 그리드에서 제공되는 컨트롤이며 아래와 같은 종류가 있습니다.

- TextEditor - 한 줄 텍스트 편집기
- MultiLineEditor - 여러 줄 텍스트 편집기
- DropDownEditor - 드롭다운 선택 편집기
- NumberEditor - 숫자 편집기
- DateEditor - 날짜 편집기
- SearchEditor - 부분검색용 편집기

그리드의 값을 가져오거나 넣기위해 사용할 수 있는 함수

Methods	GridBase	GridView	DataProvider	LocalDataProvider

Methods	GridBase	GridView	DataProvider	LocalDataProvider
GET	getValue()			getValue()
GETs		getValues()	getDistinctValues() getValues()	getFieldValues()
SET	setValue()			setValue()
SETs	setValues()			

셀의 값의 가져오거나 넣는 함수는 `itemIndex`를 인자로 사용하여 행 정보를 가져오는 함수와 `RowId`를 인자로 행 정보를 가져오는 함수로 나누어 지는데, 당연하게도 GridBase와 GridView는 `itemIndex`를 인자로 사용하고 DataProvider와 LocalDataProvider는 `RowId`를 행 정보를 가져오기 위한 인자로 사용합니다.

이 두 Index의 차이는 실제 업무화면 구현시 항상 혼란을 가져오는 요소중 하나입니다. 값을 가져오거나 넣기 위해 GridView를 사용하는지 DataProvider를 사용하는지만 명확히 구분한다면 실수를 막을수 있습니다. 주의하여 사용해야 합니다.

실습

1. 자료형을 날짜 또는 숫자로 지정하기 위해 필드의 dataType속성을 `number` 또는 `dateTime`으로 작성합니다.

```
...
{
    fieldName: "field4",
    dataType: "datetime"
},
{
    fieldName: "field5",
    dataType: "number"
},
...
```

2. `성별`컬럼의 편집기를 선택상자로 하기위해 아래와 같이 컬럼의 editor속성에 type을 `dropDown`으로 작성하고 `values`, `labels`등 속성에 값을 입력합니다.

```
...
,
{
    name: "col2",
    fieldName: "field2",
    header : {
        text: "성별"
    },
    editor : {
        type: "dropDown",
        dropDownCount: 2,
        values: ["남자", "여자"],
        labels: ["남", "여"]
    },
    width: 50
},
...
```

3. `생일`컬럼의 편집기를 날짜선택상자로 하기위해 아래와 같이 컬럼의 editor속성에 type을 `date`으로 작성하고 `datetimeFormat`속성에 값을 입력합니다.

```
...
,
{
    name: "col4",
    fieldName: "field4",
    header : {
        text: "생일"
    },
    editor: {
        type: "date",
        datetimeFormat: "yyyy-MM-dd"
    },
    width: 90
},
...
```

4. `수학`컬럼의 편집기를 숫자입력 편집기로 하기위해 아래와 같이 컬럼의 editor속성에 type을 `number`으로 작성하고 `styles`속성에 오른쪽 정렬을 위한 `textAlignment`값을 입력합니다.

```
...
,
{
    name: "col5",
    fieldName: "field5",
    header : {
        text: "수학"
    },
    editor : {
        type: "number"
    },
    styles: {
        textAlignment: "far"
    },
    width: 80
},
...
```

5. 버튼을 클릭하면 itemIndex 3, fieldIndex 2인 셀의 값을 가져오기 위해 GridBase의 getValue()함수를 호출하도록 코드를 작성 합니다.

```
//itemIndex 3, fieldIndex 2인 셀의 값 가져오기
$("#btnGetValue").click(function(){
    var value = gridView.getValue(3, 2);
    alert(value);
})
```

6. 버튼을 클릭하면 RowId 3, fieldIndex 2인 셀의 값을 '태연'으로 변경하기 위해 LocalDataProvider의 setValue()함수를 호출하도록 코드를 작성 합니다.

```
//RowId 3, fieldIndex 2인 셀에 '태연' 값을 넣기
$("#btnSetValue").click(function(){
    dataProvider.setValue(3, 2, '태연');
})
```

A26 이벤트 이해하기(Events)

Aug 17, 2015

RealGridJS RealGrid 이벤트 event

들어가며

이번 강좌에서는 이벤트에 대한 기본적인 이해를 돕기 위한 예제를 실습해 보겠습니다.

이론

RealGrid는 각각의 객체가 가지고 있는 많은 이벤트가 있습니다. 이벤트는 사용자가 그리드를 클릭하거나 값을 입력하는등 동작을 수행할때 그리드가 호출해 주는 콜백 함수 입니다.

이번 강좌에서는 GridView객체의 영역을 클릭(Click) 또는 더블클릭(Double Click) 할때 발생하는 이벤트 콜백함수를 구현하고 클릭된 인자의 정보를 팝업하는 예제를 구현해 보겠습니다.

실습

1. GridView의 영역별 클릭 또는 더블클릭 이벤트 콜백 함수를 구현합니다.

```
//log를 alert하는 함수
function showLog(log){
    alert(log);
}

gridView.onColumnHeaderClicked = function (grid, column) {
    showLog("onColumnHeaderClicked: " + "(" + column.name + ")");
};
gridView.onColumnHeaderDblClicked = function (grid, column) {
    showLog("onColumnHeaderDblClicked: " + "(" + column.name + ")");
};
gridView.onColumnCheckedChanged = function (grid, column, checked) {
    showLog("onColumnCheckedChanged: " + "(" + column.name + ", " + checked + ")")
};
gridView.onDataCellClicked = function (grid, index) {
    showLog("onDataCellClicked: " + JSON.stringify(index));
};
gridView.onDataCellDblClicked = function (grid, index) {
    showLog("onDataCellDblClicked: " + JSON.stringify(index));
};
gridView.onFooterCellClicked = function (grid, column) {
    showLog("onFooterCellClicked : " + "(" + column.name + ")");
};
gridView.onFooterCellDblClicked = function (grid, column) {
    showLog("onFooterCellDblClicked : " + "(" + column.name + ")");
};
gridView.onItemChecked = function (grid, itemIndex, checked) {
    showLog("onItemChecked: " + itemIndex + ", " + checked);
};
gridView.onItemsChecked = function (grid, items, checked) {
    showLog("onItemsChecked: " + items + ", " + checked);
```

```
};
gridView.onItemAllChecked = function (grid, checked) {
    showLog("onItemAllChecked: " + checked);
};
gridView.onCheckBarFootClicked = function (grid) {
    showLog("onCheckBarFootClicked");
};
gridView.onIndicatorCellClicked = function (grid, index) {
    showLog("onIndicatorCellClicked : " + "(" + index + ")");
};
gridView.onStateBarCellClicked = function (grid, index) {
    showLog("onStateBarCellClicked : " + "(" + index + ")");
};
gridView.onRowGroupHeadFootClicked = function (grid, index) {
    showLog("onRowGroupHeadFootClicked : " + "(" + index + ")");
};
gridView.onRowGroupHeaderFooterClicked = function (grid, index) {
    showLog("onRowGroupHeaderFooterClicked : " + "(" + index + ")");
};
gridView.onRowGroupBarClicked = function (grid, index) {
    showLog("onRowGroupBarClicked : " + "(" + index + ")");
};
gridView.onCheckBarFootDblClicked = function (grid) {
    showLog("onCheckBarFootDblClicked");
};
gridView.onIndicatorCellDblClicked = function (grid, index) {
    showLog("onIndicatorCellDblClicked : " + "(" + index + ")");
};
gridView.onStateBarCellDblClicked = function (grid, index) {
    showLog("onStateBarCellDblClicked : " + "(" + index + ")");
};
gridView.onRowGroupHeadFootDblClicked = function (grid, index) {
    showLog("onRowGroupHeadFootDblClicked : " + "(" + index + ")");
};
gridView.onRowGroupHeaderFooterDblClicked = function (grid, index) {
    showLog("onRowGroupHeaderFooterDblClicked : " + "(" + index + ")");
};
gridView.onRowGroupBarDblClicked = function (grid, index) {
    showLog("onRowGroupBarDblClicked : " + "(" + index + ")");
};
gridView.onPanelClicked = function (grid, index) {
    showLog("onPanelClicked : " + "(" + index + ")");
};
gridView.onPanelDblClicked = function (grid, index) {
    showLog("onPanelDblClicked : " + "(" + index + ")");
};
gridView.onRowGroupPanelClicked = function (grid, column) {
    showLog("onRowGroupPanelClicked : " + "(" + column.name + ")");
};
gridView.onRowGroupPanelDblClicked = function (grid, column) {
    showLog("onRowGroupPanelDblClicked : " + "(" + column.name + ")");
};
```

A27 옵션(Options)의 종류와 설정 방법

Aug 20, 2015

RealGridJS RealGrid 옵션 option

들어가며

이번 강좌에서는 RealGrid가 가지고 있는 옵션에는 어떤 것들이 있으며 기본적인 옵션 설정 방법에 대해 배워보겠습니다.

이론

RealGrid에는 많은 종류의 옵션이 있기 때문에 요구사항에 맞게 다양한 옵션을 선택하여 그리드의 설정을 변경할 수 있습니다.

옵션의 종류

1. **GridBase**
    - GridOptions : 그리드 기본 옵션
        - getOptions()
        - setOptions()
    - CopyOptions : 셀의 값을 클립보드로 복사하는 기능 관련 옵션
        - getCopyOptions()
        - setCopyOptions()
    - DisplayOptions : 그리드 UI 관련 옵션
        - getDisplayOptions()
        - setDisplayOptions()
    - EditOptions : 편집 기능 관련 옵션
        - getEditOptions()
        - setEditOptions()
    - FilteringOptions : 필터링 기능 관련 옵션
        - FilterSelectorOptions : 필터의 선택상자 관련 옵션
        - ToastOptions : 정렬/필터링/그룹핑시 토스트 메시지 옵션
        - getFilteringOptions()
        - setFilteringOptions()
    - FixedOptions : 행, 열 고정 기능 관련 옵션
        - getFixedOptions()
        - setFixedOptions()
    - GroupingOptions : 행 그룹핑 기능 관련 옵션
        - ToastOptions
        - getGroupingOptions()
        - setGroupingOptions()
    - PasteOptions : 클립보드의 값을 셀에 붙여넣기 기능 관련 옵션
        - getPasteOptions()

        - setPasteOptions()
    - SelectOptions : 셀 선택 기능 관련 옵션
        - getSelectOptions()
        - setSelectOptions()
    - SortingOptions : 정렬 기능 관련 옵션
        - ToastOptions
        - getSortingOptions()
        - setSortingOptions()
2. **TreeView**
    - TreeOptions : 트리 기본 옵션
        - getTreeOptions()
        - setTreeOptions()
3. **DataProvider**
    - DataProviderOptions : DataProvider 기본 옵션
        - getOptions()
        - setOptions()
4. **Component**
    - Panel
        - getPanel()
        - setPanel()
    - Header
        - getHeader()
        - setHeader()
    - Indicator
        - getIndicator()
        - setIndicator()
    - StateBar
        - getStateBar()
        - setStateBar()
    - CheckBar
        - getCheckBar()
        - setCheckBar()
    - Footer
        - getFooter()
        - setFooter()
5. **함수의 인자로 사용되는 옵션**
    - DataFillOptions (RealGridJS only)
        - fillCsvData()
        - fillJsonData()
        - fillXmlData()
    - DataLoadOptions (RealGrid+ only)
        - loadData()

- GridExportOptions
  - exportGrid()
- RowGroupOptions
  - setRowGroup()
- SearchOptions
  - searchCell() (RealGridJS only)
  - searchItem()

실습

1. 보이기, 감추기 버튼을 누를때 panel, indicator, checkbar를 보이거나 감추도록 옵션 설정 코드를 작성합니다.

```
// panel, indicator, checkbar 감추기
$("#btnSetHide").on("click", function(){
    gridView.setPanel({
        visible: false
    });

    gridView.setIndicator({
        visible: false
    });

    gridView.setCheckBar({
        visible: false
    });
})

// panel, indicator, checkbar 보이기
$("#btnSetShow").on("click", function(){
    gridView.setPanel({
        visible: true
    });

    gridView.setIndicator({
        visible: true
    });

    gridView.setCheckBar({
        visible: true
    });
})
```

Github

Help

RSS