

## Обучение Ext JS :: Часть 5. Информация в таблице

Фев 21 2010Обращение GridPanel

GridPanel, это та штука, которая держит все вместе, заставляет работать и заботится о размещении данных по столбцам и рядам, подмывает заголовков столбцов и собирает все это в аккуратную таблицку.

Информация по фильмам не принесет никакой пользы, если будет просто сидеть в памяти компьютера. Давайте выведем ее через таблицу:

1. Добавьте ваше хранилище данных в код GridPanel:

Исходный код примера:

## Исходный код примера:

```
Ext.onReady(function(){
    // add your data store here
    var grid = new Ext.grid.GridPanel({
        renderTo: document.body,
        frame:true,
        title: 'Movie Database',
        height:200,
        width:500,
        store: store,
        columns: [
            {header: 'Title', dataIndex: 'title'},
            {header: 'Director', dataIndex: 'director'},
            {header: 'Released', dataIndex: 'released',
            },
            {header: 'Ext.util.Format.dateRenderer('m/d/Y')',
            },
            {header: 'Genre', dataIndex: 'genre'},
            {header: 'Tagline', dataIndex: 'tagline'}
        ]
    });
});
```

2. Загрузите в браузере и вы увидите примерно следующее:

Movie Database				
Title	Director	Released	Genre	Tagline
Office Space	Mike Judge	02/19/1999	1	Work Sucks
Super Troopers	Jay Chandrasekhar	02/15/2002	1	Altered State Police
American Beauty	Sam Mendes	10/01/1999	2	... Look Closer
The Big Lebowski	Joel Coen	03/06/1998	1	The "Dude"
Fight Club	David Fincher	10/15/1999	3	How much can you

Как это работает?

Наше хранилище было отправлено в таблицу вместе с моделью столбца, что определяет то, как будут отображаться столбцы и их заголовки. Это отличается от использования поля данных обрабатываемых считывателем, которое было настроено так, чтобы считыватель знал как читать данные.

Настройка GridPanel

GridPanel является приложением которая связывает все воедино. :

Ext.grid.GridPanel

Для настройки GridPanel нам необходимо несколько основных элементов:

Тип поля	Описание	Применение
renderTo	Где должна отображаться таблица?	Это должно быть объектом DOM, или ID компонента DOM. Позже мы будем отправлять GridPanel на обработку сразу в другие приложения. Следовательно эта настройка скоро устарееет.
рамки	Обрамляют таблицу	Это просто создает милую границу и вокруг GridPanel, а так же добавляет панель названия. необязательно, но при обработке страницы выглядит здорово.
Высота и ширина	размер в пикселях	Высота, при использовании таблицы, нужна почти всегда, так как таблица сама не может определить свою высоту. Когда мы начнем использовать макеты таблицы это больше нам не потребуется.
Хранилище	Наши данные	Это ссылка на действительное хранилище данных, где проживают наши данные.
stripeRows	Ряды полос	Используется для изменения цветов в рядах данных

Основные настройки для нашей панели будут выглядеть вот так:

Исходный код примера:

## Исходный код примера:

```
var grid = new Ext.grid.GridPanel({
    renderTo: Ext.getBody(),
    frame:true,
    title: 'Movie Database',
    height:200,
    width:500,
    store: store,
    columns: [ insert columns here ]
});
```

И мы как всегда можем прочитать их как предложение:

Вставь нашу таблицу в страницу, создай для нее границы и назови 'Movie Database'. Высота должна быть 200 а ширина 500; Она будет использовать наше хранилище 'store' и у нее будут указанные столбцы.

Существует одна причина, по которой я люблю настроить основывающиеся на объекте. И эта причина заключается в том, что их можно прочитать. Нам никогда не придется вручную искать какой аргумент 3 у функции x. Мы просто говорим "Сделай высоту 200 и ширину 500".

Определяем модель столбца таблицы

Для этого нам необходимо создать массив объектов, который определит как отображать столбцы и вообще что с ними делать.

```
columns: [
    {header:
    {header:
    {header:
    {header:
```

Это создаст заголовок таблицы, который будет выглядеть примерно вот таким образом:

Title	Director	Released	Genre	Tagline
-------	----------	----------	-------	---------

У объекта, определяющего каждый столбец может быть множество настроек, но нам пока требуется чтобы были указаны заголовок и dataIndex. Настройки заголовка, это просто текст, выводимый в нем. А настройками dataIndex является название поля данных, используемое в этом столбце. Мы указываем их когда настраиваем считыватель хранилища.

ниже привожу несколько полезных настроек для модели столбца

Настройка	Описание	Использование
render	Указывает как именно должны отображаться данные	Может быть использован для преобразования данных для столбца в необходимый вам формат. Работает со всеми типами данных.
hidden	Прячет столбец	Логическое значение, определяющее отображать или не отображать столбец.
width	Указывает ширину столбца в пикселях	Ширина столбца. По умолчанию стоит 100 пикселей. Переполняющее содержимое скрывается.
sortable	Указывает можно ли применять к столбцу сортировку	Логическое значение, указывающее на наличие возможности отсортировать содержимое столбца.

Использование обработки ячеек

При помощи обработки ячеек, мы можем совершать довольно хитрые штуки. Есть несколько ограничений, удерживающие нас от того чтобы ячейка содержала все что нам заблагорассудится, но их очень мало. Все что нужно сделать, это указать одну из встроенных форматирующих ячеек функций Ext JS, например usMoney, или создать собственный обработчик ячеек, который будет возвращать преобразованные значения.

Сначала мы посмотрим как работает встроенный обработчик, после чего попробуем создать наш собственный.

Преобразование данных с использованием встроенного обработчика ячейки

Большинство встроенных преобразующих функций существуют для того, чтобы предоставить быстрый доступ к самым распространенным преобразующим функциям. Чаще всего, я использую обработчик данных:

```
renderer: Ext.util.Format.dateRenderer('m/d/Y')
```

Некоторые другие рендеры включают часто-требуемое форматирование: денежное, верхнего и нижнего регистровSome other renderers include some commonly-required formatting, such as money, capitalize, and lowercase.

Еще несколько обработчиков, которые могут быть полезны:

Обработчик	Описание	Использование
dateRenderer		
uppercase	Конверсия верхнего и нижнего регистров	Преобразует строку в текст другого регистра.
lowercase		
capitalize	карсивый текст	Правильно расставляет заглавные буквы.

Создание поисковых хранилищ данных или самостоятельная обработка ячеек

Мы начнем с того, что возьмем столбец "Жанр", у которого есть числовое значение и найдем это значение в нашей базе, созданной в главном по формам для того чтобы найти текстовое отображение номера нашего жанра.

Сначала, мы добавим настройки в модель столбца, которые будут подсказывать какую функцию использовать для обработки содержимого ячейки.

```
{header: 'Genre', dataIndex: 'genre', renderer: genre_name}
теперь давайте создадим эту функцию. Вызову функции в качестве первого аргумента присваивается значение ячейки. Вторым аргументом является объект ячейки, в то время как третий, это данные хранимые для таблицы. Ничего из этого мы для обработки использовать не будем, так что просто оставим..
```

Исходный код примера:

## Исходный код примера:

```
function genre_name(val){
    return genres.queryBy(function(rec){
        if (rec.data.id == val){
            return true;
        }else{
            return false;
        }
    });itemAt(0).data.genre;
}
```

Функция обработки передает значение текущей информации ячейки. Это значение может быть проверено и с ним можно что-нибудь сделать (какое бы значение не было бы возвращено функцией оно обработано для ячейки таблицы). Обработчик queryBy используется для фильтрации данных из нашего хранилища. Он принимает функцию, которая сравнивает каждый ряд данных и разрешает использовать тех, которые прошли проверку.

В дополнение к вышесказанному, я привожу сокращенную версию той же самой функции. Ее труднее прочитать, но она выдает тот же самый результат.

```
function genre_name(val){
    return genres.queryBy(function(rec){
        return rec.data.id == val;
    });itemAt(0).data.genre;
}
```

Совмещение двух столбцов

Мы можем совместить два столбца, так как нам не надо чтобы он отображался сразу в двух местах. Спрятать столбец можно в нашей модели столбца:

```
{header: 'Tagline', dataIndex: 'tagline', hidden: true}
Следующим шагом будет наша функция рендера, которая позаботится об объединении столбцов.
```

```
function title_tagline(val, x, store){
    return '<b>'+val+'<b><b><b>'+store.data.tagline;
}
```

И пошел дальше и выделил название жирным шрифтом для того, чтобы немного различать два элемента. Как вы видите теги HTML отлично работают в таблице ячеек. Следующим шагом будет обработка настроек модели столбца, приделывая свеже созданную функцию title\_tagline.

```
{header: 'Title', dataIndex: 'title', renderer: title_tagline}
```

После этого наш столбец будет выглядеть таким образом:

Title	Director	Released	Genre	Price
Office Space	Mike Judge	02/19/1999	Comedy	\$19.95
Work Sucks				

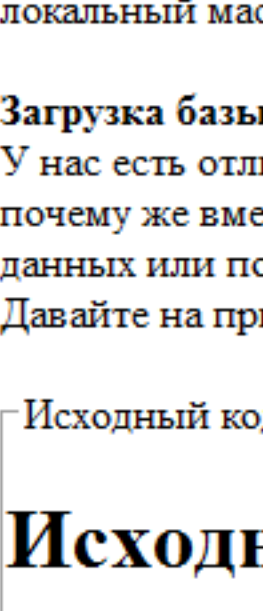
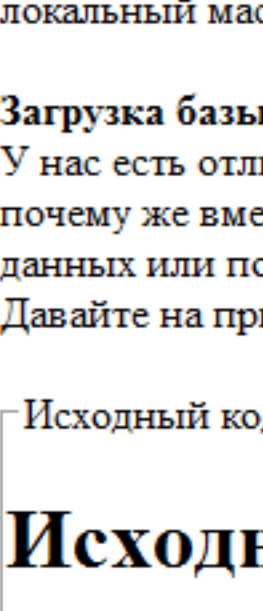
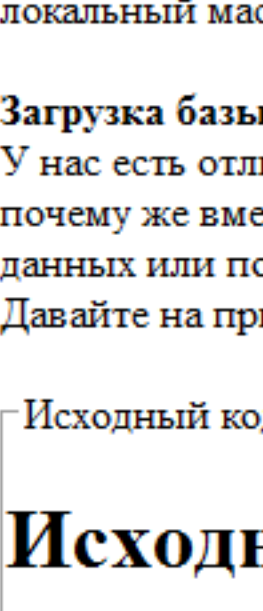
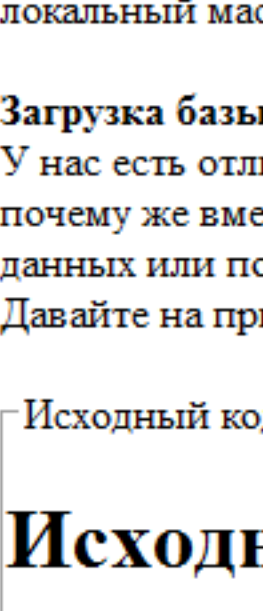
Создание HTML и графикс

Добавим эрештливости, разместив в каждый ряд по изображению, скажем, обложку фильма, как только что стало известно, мы можем в ячейке использовать простой HTML. Таким образом, все что от нас требуется, это создать рендерер который будет брать поле содержимое

название файла изображения и прописывать его в тег IMG как характеристику SRC.

```
function cover_image(val){
    return '<img src=images/'+val+'>';
}
```

С этой, довольно незамысловатой функцией, и настроенным рендерером столбцов, мы получаем картинку в нашу таблицу. Если вы все сделали правильно, то таблица должна выглядеть таким образом:

Cover	Title	Director	Released	Genre	Price
	Office Space	Mike Judge	02/19/1999	Comedy	\$19.95
	Work Sucks				
	Super Troopers	Jay Chandrasekhar	02/15/2002	Comedy	\$14.95
	Altered State Police				

Встроенные свойства

У Ext есть некоторые, по настоящему хорошие, встроенные свойства, помогающие закончить табличный интерфейс. Столбцы обладают встроенными меню, позволяющим сортировать, отображать и прятать столбцы.

Movie Database				
Title	Director	Released	Genre	
Office Space	Mike Judge	Fri Feb-19 1999 00: 19:95	1	Work Sucks
Super Troopers	Jay Chandrasekhar	Fri Feb-15 2002 00: 14:95	1	Altered State Police
American Beauty	Sam Mendes	Fri Oct 01 1999 00: 19:95	2	... Look Closer
The Big Lebowski	Joel Coen	Fri Mar 06 1998 00: 21:9	1	The "Dude"
Fight Club	David Fincher	Fri Oct 15 1999 00: 19:95	3	How much can you

Сортировка с клиентской стороны

До тех пор, пока не указана возможность сортировать таблицу с серверной стороны, она будет сортировать столбцы с клиентской. Серверной сортировкой стоит пользоваться тогда, когда данные разбиты на страницы или находятся в таком формате, что клиентская сортировка невозможна. Клиентская сортировка быстра, проста и встроена:

```
{header: 'Tagline', dataIndex: 'tagline', sortable: true}
```

Мы также можем сделать это после того, как таблица была обработана:

```
var colmodel = grid.getColumnModel();
colmodel.setColumById('tagline', true);
```

Наша модель столбца управляет отображением столбцов и заголовков. Если мы возьмем ссылку на модель столбца спросив ее у формы таблицы, тогда мы сможем сделать изменения в столбце после того как он был обработан. Это можно сделать при помощи обработчика setColumById который нам предоставит модель столбца и который принимает ID столбца как аргумент.

Скрытие/Видимые столбцы

Справочное хранилище обычно заголовка столбца могут быть видимыми и скрытыми. Так же это можно изменить на уровне настроек, прописав в свойствах скрытность по умолчанию:

```
{header: 'Tagline', dataIndex: 'tagline', hidden: true}
```

Еще один способ сделать это после того как таблица была обработана - использовать функцию,которую предоставляет Ext

```
Var colmodel = grid.getColumnModel();
colmodel.setHidden(colmodel.getId('tagline'), true);
```

Присвоение ссылки модели столбца снова позволит нам сделать эти изменения.

Изменения порядка столбцов

Перетаскивание пользователем заголовка столбца, позволит ему выстроить все столбцы в требуемом порядке. Все это по умолчанию включено во встроенный функционал таблицы.

Movie Database				
Title	Director	Released	Price	Tagline
Office Space	Mike Judge	Fri Feb-19 1999 00: 19:95	1	Work Sucks
Super Troopers	Jay Chandrasekhar	Fri Feb-15 2002 00: 14:95	1	Altered State Police
American Beauty	Sam Mendes	Fri Oct 01 1999 00: 19:95	2	... Look Closer
The Big Lebowski	Joel Coen	Fri Mar 06 1998 00: 21:9	1	The "Dude"
Fight Club	David Fincher	Fri Oct 15 1999 00: 19:95	3	How much can you

Любой столбец может быть перенесен на другое место в таблицу. Этот скриншот демонстрирует процесс переноса столбца "Цены" между столбцами "Название" и "Режиссер".

Мы можем полностью отключить эту функцию, выставив соответствующие настройки в GridPanel:

enableColumnMove: false

Настройка disableColumnMove: false (и не только их) можно отслеживать и указать что должна быть реакция. Например мы можем отслеживать перемещения столбца, и выдавать следующее сообщение о том, куда он был перемещен:

Исходный код примера:

## Исходный код примера:

```
grid.getColumnModel().on('columnmoved',
function(cm,indx,nindx) {
    var title = 'You Moved '+cm.getColumnHeader(nindx);
    if (indx > nindx){
        var dirmsg = (indx-nindx)+' Column(s) to the Left';
    }else{
        var dirmsg = (nindx-indx)+' Column(s) to the Right';
    }
    Ext.Msg.alert(title,dirmsg);
});
```

Отображение серверных данных в таблице

При помощи Ext мы можем извлекать данные на нашу таблицу самыми разными способами. Мы начали с того, что использовали локальный массив и использовали его в таблице. Теперь мы извлечем данные из внешнего файла и веб-сервера.

Загрузка базы данных их файла XML

У нас есть отличная база данных по фильмам, но каждый раз, добавляя новый, нам приходится редактировать массив JavaScript. Так почему же вместо этого, не хранить базу в формате XML? Ее будет легче обновлять, а файл XML может быть создан из запроса базы данных или пользователяского скрипта

Давайте на примере рассмотрим как будет размечен наш файл XML:

Исходный код примера:

## Исходный код примера:

```
<?xml version="1.0" encoding="UTF-8"?>
<dataset>
<row>
    <id>1</id>
    <title>Office Space</title>
    <director>Mike Judge</director>
    <released>1999-02-19</released>
    <genre>1</genre>
    <tagline>Work Sucks</tagline>
    <coverthumb>84m.jpg</coverthumb>
    <price>19.95</price>
    <active>1</active>
</row>
<row>
    <id>3</id>
    <title>Super Troopers</title>
    <director>Jay Chandrasekhar</director>
    <released>2002-02-15</released>
    <genre>1</genre>
    <tagline>Altered State Police</tagline>
    <coverthumb>42m.jpg</coverthumb>
    <price>14.95</price>
    <active>1</active>
</row>
<!--more rows of data removed for readability--!>
</dataset>
```

Еще, нам предстоит изменить считыватель данных, установить местоположение нашего файла XML так, чтобы хранилище знало откуда брать данные.

При переходе с локальных данных на удаленные, необходимо сделать четыре изменения:

• Надо доавить настройку url, указывающей местоположение наших данных. Это изменит настройки данных, которые мы использовали при ее хранении на локальных ресурсах.

• Считыватель меняется с AjaxReader на XmlReader для того чтобы разобраться с различиями, связанными с форматами.

• Указать XmlReader какой именно компонент содержит запись ряда, устанавливая настройки запроса.

• Необходимо указать запрос для загрузкии функции, говорящей нашему хранилищу когда передавать данные в память

Исходный код примера:

## Исходный код примера:

```
var store = new Ext.data.Store({
    url: 'movies.xml',
    reader: new Ext.data.XmlReader({
        record:'row',
        id:'id'
    }, [
        {id:
        'coverthumb',
        'title',
        'director',
        {name: 'released', type: 'date', dateFormat: "Y-m-d"},
        'genre',
        'tagline',
        {name: 'price', type: 'float'},
        {name: 'available', type: 'bool'}
        ]
    });
store.load();
```

После внесения этих изменений вы заметите существенную разницу вызванную сменой источника данных и форматов. Обратите внимание, что для того чтобы заменить локальные данные на удаленные, и перейти от массива к формату XML мы внесли изменения только в хранилище. Ext выделяет эти изменения используя общее хранилище, которое может пользоваться внешним считывателем и применять его для многих форматов.

Загрузка базы данных по фильмам из фала JSON

Это тот же самый случай что и с XML. Просто измените считыватель и поменяйте кое-какие настройки.

Предполагается, что ряды данных JSON находятся в виде массива объектов. Наш файл movies.json будет содержать следующую информацию:

Исходный код примера:

## Исходный код примера:

```
{
    success:true,
    rows:[
    {
        "id":"1",
        "title":"Office Space",
        "director":"Mike Judge",
        "released":"1999-02-19",
        "genre":"1",
        "tagline":"Work Sucks",
        "coverthumb":"84m.jpg",
        "price":"19.95",
        "active":"1"
    },
    {
        "id":"3",
        "title":"Super Troopers",
        "director":"Jay Chandrasekhar",
        "released":"2002-02-15",
        "genre":"1",
        "tagline":"Altered State Police",
        "coverthumb":"42m.jpg",
        "price":"14.95",
        "active":"1"
    }
    ],
    //...more rows of data removed for readability...!>
}
```

Основное различие между считывателями JSON и XML заключается в том, что в случае с JSON, считывателю необходимо знать название корневого компонента, содержащего массив объектов (информация). Таким образом, вместо указания настроек записи мы указываем корневые настройки:

Исходный код примера:

## Исходный код примера:

```
var store = new Ext.data.Store({
    url: 'movies.json',
    reader: new Ext.data.JsonReader({
        root:'rows',
        id:'id'
    }, [
        {id:
        'coverthumb',
        'title',
        'director',
        {name: 'released', type: 'date', dateFormat: "Y-m-d"},
        'genre',
        'tagline',
        {name: 'price', type: 'float'},
        {name: 'available', type: 'bool'}
        ]
    });
store.load();
```

Эта таблица будет выглядеть точно так же, и иметь тот же функционал что массив и таблицы XML, созданные нами ранее.

**ВАЖНО:** *JSON является родным форматом JavaScript, а значит что данные будут обрабатываться быстрее, соответственно, загрузка таблицы будет занимать меньше времени.*

Загрузка данных из базы используя PHP

Настройку нашей GridPanel можно оставить без изменений. Но вместо того чтобы брать статичный файл с данными JSON, мы можем воспользоваться скриптом PHP, который будет проводить выборку из базы и форматировать ее в понятный для Ext формат JSON:

Исходный код примера:

## Исходный код примера:

```
<?php
// connect to database
$sql= "SELECT * FROM movies";
$arr = array();
If (($rs = mysql_query($sql)) {
    Echo '{success:false}';
}else{
    while($obj = mysql_fetch_object($rs)){
        $arr[] = $obj;
    }
    Echo '{success:true;rows:json_encode($arr)}';
}>
```

**ВАЖНО:** Код PHP который использован в этом примере содержит самый минимум того, что необходимо для завершения работы. В реальных условиях вы непременно захотите добавить настроек безопасности, проверку на ошибки и многое другое, но на этом учебный код не расписан.

- « первая
- « предыдущая
- 1
- 2
- 3
- « следующая »
- « последняя »
- English