Главная › Блоги › Lertion's блог

[Обучение Ext JS] :: Часть 5. Информация в таблице

Фев 21 2010**Программируем таблицу**
Большая часть написанного нами кода настраивает порядок появления элементов таблицы. Довольно часто хочется сделать что-нибудь, что заставило бы таблицу реагировать на действия пользователя. Одна из наиболее используемых функций это выбор и перемещение рядов данных. **Ext JS** поддерживает такое мракобесие и называется это "модель выбора". Давайте попробуем ее настроить.

Работая с выбором ячейки и ряда, таблица **Ext** позволяет отслеживать действия пользователя при помощи модели выбора. Эта модель используется для определения того, каким образом выделены элементы, сколько и какие именно. Можно создать слушателей для этих событий которые будут сообщать всю интересующую нас информацию.

Некоторые из моделей выбора:
- CellSelectionModel: Позволяет пользователю выбирать одну ячейку из таблицы.
- RowSelectionModel: Позволяет выбирать весь ряд.
- ColumnSelectionModel: Позволяет выбирать весь столбец
- CheckBoxSelectionModel: Это выдает чекбокс, который делает возможным выбор рядов.

Выбор модели целиком и полностью зависит от требований вашего проекта. Для нашей базы фильмов мы используем распространенную модель выбора ряда.

Модель выбора определяется в настройках GridPanel используя это:

```
sm: new Ext.grid.RowSelectionModel({
  singleSelect: true
})
```

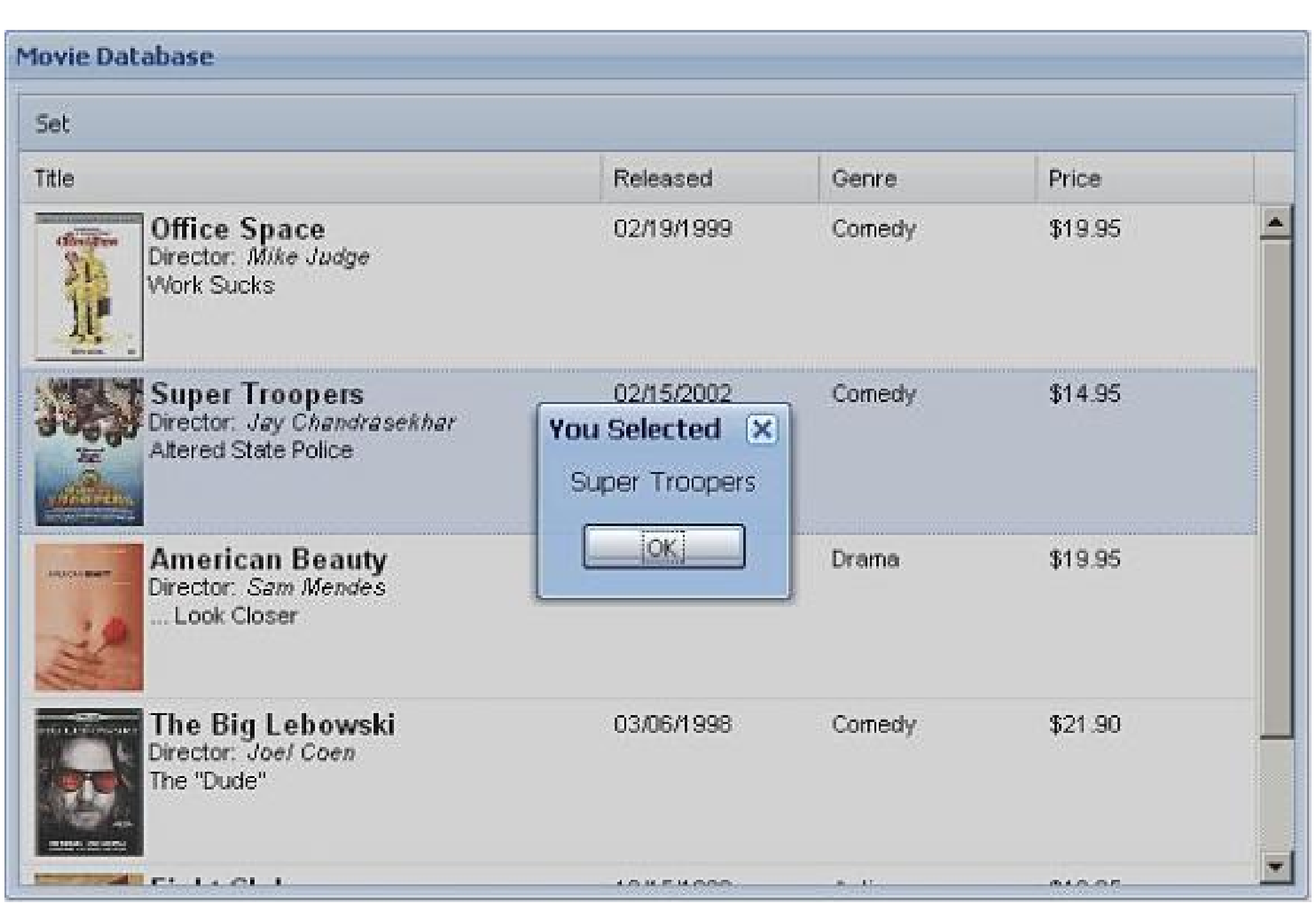
Мы, так же, снабдим модель выбора настройками, которые будут отвечать за выбор только одного ряда. Это не даст пользователю выбрать сразу несколько позиций.

Слушаем нашу модели на предмет выборов
Слушатели таблицы могут быть включены во многих местах, в зависимости от желаемого взаимодействия. Ранее, мы применили слушатель к столбцу, поскольку мы хотели знать о действиях с ним.
Здесь мы добавим слушатель в модель выбора потому что мы хотим знать когда пользователь выберет фильм.

Исходный код примера:

Исходный код примера:

```
sm: new Ext.grid.RowSelectionModel({
  singleSelect: true,
 listeners: {
    rowselect: {
      fn: function(sm,index,record) {
        Ext.Msg.alert('You Selected',record.data.title);
      }
    }
  }
})
```



Title	Released	Genre	Price
Office Space Director: *Mike Judge* Work Sucks	02/19/1999	Comedy	$19.95
Super Troopers Director: *Jay Chandrasekhar* Altered State Police	02/15/2002	Comedy	$14.95
American Beauty Director: *Sam Mendes* ... Look Closer		Drama	$19.95
The Big Lebowski Director: *Joel Coen* The "Dude"	03/06/1998	Comedy	$21.90

Теперь выбор ряда вызывает окно предупреждения. Давайте посмотрим что тут происходит:
- Слушатель выставлен на событие rowselect. Он ждет до тех пор пока не будет выбран ряд, и затем выполняет функцию.
- Наша функция обращается к модели выбора, получает метку ряда и информацию записанную в этом самом ряде.
- Используя полученные данные мы берем название фильма вставляем его в диалоговое окно.

Управление таблицей при помощи кода
Существует много функция позволяющих управлять таблицей и содержащейся в ней информацией. Их можно привязать к другим приложениям Ext для создания практически любого функционала.

Изменение таблицы нажатием кнопки
Сейчас мы собираемся добавить в нашу таблицу панель инструментов, на которой будут кнопки вызывающие строку, позволяющую изменять название фильма.

Исходный код примера:

Исходный код примера:

```
tbar: [{
  text: 'Change Title',
  handler: function(){
    var sm = grid.getSelectionModel();
    if (sm.hasSelection()){
      var sel = sm.getSelected();
      Ext.Msg.show({
      title: 'Change Title',
        prompt: true,
        buttons: Ext.MessageBox.OKCANCEL,
        value: sel.data.title,
        fn: function(btn,text){
          if (btn == 'ok'){
            sel.set('title', text);
          }
        }
      });
    }
  }
}]
```

Title	Released	Genre	Price
Office Space Director: *Mike Judge* Work Sucks	02/19/1999	Comedy	$19.95
Super Troopers Director: *Jay Chandra* Altered State Police			$14.95
American Beauty Director: *Sam Mendes* ... Look Closer	10/01/1999	Drama	$19.95

Все что мы тут делаем, так это изменяем данные в хранилище. Данные нашей базы на сервере остались точно такими же, а сервер вообще не имеет понятия о том, что что-то изменилось. Лишь от нас зависит каким образом донести до сервера эти изменения, или через **AJAX**-запрос или как вам там еще удобно.

Что же будет тут происходить?:
- sm: Мы получаем модель выбора из таблицы
- sel: Используем модель для получения выбранного ряда
- sel.data: Используя объект данных выбранного элемента мы получаем данные

При помощи этого несложного способа можно создать много смешных взаимодействий. Единственные ограничения это 24 часа в сутках и необходимость сна.

Продвинутое форматирование таблицы
Раз уж мы собираемся создать несколько взаимодействий, так почему бы не добавить несколько кнопок которые будут делать всякие смешные штуки. К примеру эта кнопка позволяющая спрятать столбец. Мы также изменим текст кнопки, основываясь на видимости столбца:

Исходный код примера:

Исходный код примера:

```
{
 text: 'Hide Price',
 handler: function(btn){
   var cm = grid.getColumnModel();
   var pi = cm.getIndexById('price');
   if (cm.isHidden(pi)){
     cm.setHidden(pi,false);
     btn.setText('Hide Price');
   }else{
     cm.setHidden(pi,true);
     btn.setText('Show Price');
   }
   btn.render();
 }
}
```

Здесь мы использовали новый обработчик — getIndexById, который, как вы уже поняли берет метку столбца и будет на один меньше чем все количество столбцов (отсчет начинается с 0, да). Этот номер служит для определения того, в каких отношениях находится выбранный столбец со своими соседями. В коде нашей таблицы столбец "цена" является четвертым, что означает - его метка 3.

Разбиение таблицы на страницы
Разбиение на страницы требует чтобы у нас был серверный компонент (скрипт), который будет, собственно разбивать наши данные по страницам. С этогои начнем. Лучше всего для наших целей подходит **PHP**. Код легко понять и перевести в другие языки. Поэтому в нашем примере он и используется.

Когда происходит разбиение таблицы на страницы, она отправляет параметры начала и предела серверному скрипту. Это похоже на то, что используется в базе данных для выбора подмножества записей. Наш скрипт читает эти параметры и сразу же использует из в запросе к базе данных

Вот типичный скрипт **PHP** который обрабатывает разбивку на страницы. Назовем файл movies-paging.php.

Исходный код примера:

Исходный код примера:

```
<?php
// connect to database
$start = ($_REQUEST['start'] != '') ? $_REQUEST['start'] : 0;
$limit = ($_REQUEST['limit'] != '') ? $_REQUEST['limit'] : 3;
$count_sql = "SELECT * FROM movies";
$sql = $count_sql . " LIMIT ".$start.", ".$limit;
$arr = array();
If (!$rs = mysql_query($sql)) {
  Echo '{success:false}';
}else{
  $rs_count = mysql_query($count_sql);
  $results = mysql_num_rows($rs_count);
  while($obj = mysql_fetch_object($rs)){
    $arr[] = $obj;
  }
  Echo '{success:true,results:'.$results.',
      rows:'.json_encode($arr).'}';
}
?>
```

Этот скрипт PHP позаботится о серверной части разбивки. Теперь нам нужно добавить инструмент для нее (разбивки) на панель инструментов.

Title	Released	Genre	Price
American Beauty Director: *Sam Mendes* ... Look Closer	10/01/1999	Drama	$19.95
Fight Club Director: *David Fincher* How much can you know	10/15/1999	Action	$19.95
Office Space Director: *Mike Judge* Work Sucks	02/19/1999	Comedy	$19.95

Раньше мы использовали панель инструментов расположенную вверху, теперь мы поместим ее вниз (панель перехода по страницам и сверху? Выглядит нелепо).

Этим кодом мы добавим переход по страницам:

```
bbar: new Ext.PagingToolbar({
  pageSize: 3,
  store: store
})
```

И конечно нам нужно изменить url нашего хранилища данных на url серверного кода **PHP**. При разбивке также потребуется totalProperty. Это переменное название содержащее полную запись рядов в БД.

Исходный код примера:

Исходный код примера:

```
var store = new Ext.data.Store({
 url: 'movies-paged.php',
 reader: new Ext.data.JsonReader({
   root:'rows',
  totalProperty: 'results',
   id:'id'
 }, [
   // data column model removed for readability
 ])
});
```

Группировка
Группировка таблиц используется для того чтобы ряды выглядели похожими. Так же это дает нам возможность сортировать то, что принадлежит только отдельно взятой группе. Таким образом если мы хотим отсортировать столбец цены, цены будут отсортированы только в пределах каждой группы элементов.

Групповое хранилище
Нам необходимо особое хранилище которое называется GroupingStore. Его настройки схожи со стандартным хранилищем. На надо только задать несколько дополнительных настроек - sortInfo и groupField. В самой информации никаких изменений делать не надо, потому что со стороны пользователя обо всем позаботится **Ext JS**.

Исходный код примера:

Исходный код примера:

```
var store = new Ext.data.GroupingStore({
 url: 'movies.json',
 sortInfo: {
   field: 'genre',
   direction: "ASC"
 },
 groupField: 'genre',
 reader: new Ext.data.JsonReader({
  root:'rows',
  id:'id'
 }, // reader column model here //)
});
```

Нам так же необходимо добавить на панель таблицы настройки просмотра. Это позволит визуально учитывать группы.

Исходный код примера:

Исходный код примера:

```
var grid = new Ext.grid.GridPanel({
 renderTo: document.body,
 frame:true,
 title: 'Movie Database',
 height:400,
 width:520,
 store: store,
 autoExpandColumn: 'title',
 columns: // column model goes here //,
 view: new Ext.grid.GroupingView()
});
```

После всех внесенных изменений у нас должно получиться что-то вроде этого:

Title	Released	Genre ▲	Price
⊟ Genre: Comedy			
The Big Lebowski Director: *Joel Coen* The "Dude"	03/06/1998	Comedy	$21.90
Office Space Director: *Mike Judge* Work Sucks	02/19/1999	Comedy	$19.95
Super Troopers Director: *Jay Chandraseki* Altered State Police	02/15/2002	Comedy	$14.95
⊟ Genre: Drama			
American Beauty Director: *Sam Mendes* ... Look Closer	10/01/1999	Drama	$19.95

Если вы теперь раскроете контекстное меню из заголовка столбца вы увидите новый элемент Group By This Field который позволит пользователю сгруппировать столбцы на ходу.

Заключение
В этой главе мы научились многое касающееся представления данных в таблице. С этим знанием мы можем организовывать огромные количества информации в понятные и простые таблицы.

Мы рассмотрели:
- Создание хранилищ данных и таблиц для их отображения.
- Чтение данных **XML** и **JSON** получаемых с сервера и выводимых через таблицу.
- Обработку ячеек для красивого отображения данных.
- Изменение таблицы в зависимости от действий пользователя.

Мы так же обсудили сложности связанные с каждым из элементов. Мы так же рассмотрели форматирование ячеек используя **HTML**, изображения, и даже поиск в отдельно взятом хранилище данных.

Теперь,когда мы знаем о стандартных таблицах, мы готовы перейти наследующий уровень - сделать нашу таблицу доступной для редактирования, совсем как динамическая таблица. И это будет темой нашей следующей главы.

- « первая
- ‹ предыдущая
- 1
- 2
- 3

- English

Голос
28