[Обучение Ext JS] :: Часть 14. Сила Ext JS: Что еще можно сделать?

Мар 21 2010

Мы рассмотрели некоторые основные компоненты **Ext JS**: Окни и диалоги, таблицы, деревья, хранилища данных, и многое другое. Это одна из ярких и наиболее последовательных вещей в Сети для обработки старого доброго **HTML** и придания ему практически невероятные возможности для разработки приложений. Но мы всего-лишь царапнули поверхность, оставив огромное количество возможностей нераскрытыми.

Еще так много работы
Сайт **Ext JS** (http://www.extjs.com) дает огромное количество информации для тех кто учится использовать **Ext JS**. Быстрое знакомство с разделами Samples и Demos покажут почти все, чего мы коснулись в этой книге, но тщательное исследование исходного кода покажет, что есть вещи, о существовании которых мы и не знали. Они не такие явные на подобии панели и таблицы, но эти маленькие вещи поддерживают всю конструкцию и помогают создать крепкое приложение.

Виджеты формы
Что такое виджет? Ну, виджет это крошечный кусочек или компонент функционала, например ползунок прогрессбара. Большинство приложений состоят из форм, так что не удивительно, что **Ext JS** включает в себя некоторые виджеты формы, которые мы не можем получить прямо из HTML. TextFields, Checkboxes и Radio-кнопки, это стандартная плата. Но **Ext JS** умеет подсластить пилюлю переход на новый уровень, давая нам компоненты, которые мы больше привыкли видеть в десктопных приложениях.

DateField
DateField является отличным примером. Обработка дат внутри большинства форм **HTML** может быть рутиной, но **Ext JS** дает простой компонент для обработки дат:
Example 1: ch14ex1.js

Исходный код примера:

Исходный код примера:

```
Ext.onReady(function(){
  var formPanel = new Ext.form.FormPanel({
    title:'DateField Example',
    applyTo:'chap14_ex01',
    layout:'form',
    labelAlign:'top',
    width:210,
    autoHeight:true,
    frame:true,
    items:[{
    xtype:'datefield',
      fieldLabel:'Date of Birth',
      name:'dob',
      width:190,
      allowBlank:false
    }]
  });
});
```

Здесь у нас простая разметка FormPanel, размещающая наш DateField в подходящий контекст и позволяющей определенные атрибуты (как размещение таблицы в поле). Внутри разметки у нас есть единственный компонент DateField. При начальной загрузке он выглядит как простой TextField, за исключением выключателя справа. Нажатие на него показывает красоту и постоянство компонента **Ext JS**, отображая привлекательный календарь в разделе даты. Можно выбрать день, перейти от месяца к месяца или даже нажав на месяц выбрать год.

Отдельно от богатого отображения, DateField также дает полный набор атрибутов для более детальной настройки, включая возможность определять различные принятые форматы даты и возможность отключать блоки дней. Компонент так же будет проверять данные введенные вручную.

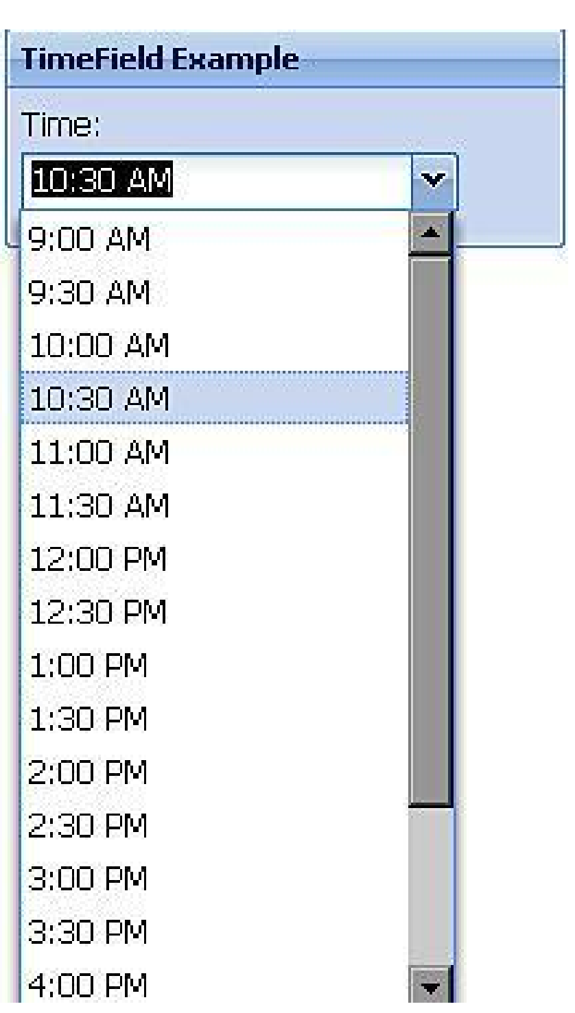
TimeField
Работа со временем может быть более сложной чем работа с датами, но и снова у **Ext JS** есть компонент чтобы помочь нам. TimeField это просто комбинированное окно, которое автоматически показывает набор времени, с нужным нам инкрементом для быстрого выбора:
Example 2: ch14ex2.js

Исходный код примера:

Исходный код примера:

```
Ext.onReady(function(){
  var formPanel = new Ext.form.FormPanel({
    title:'DateField Example',
    applyTo:'chap14_ex02',
    layout:'form',
    labelAlign:'top',
    width:210,
    autoHeight:true,
    frame:true,
    items:[{
      xtype:'timefield',
      fieldLabel:'Time',
      minValue: '9:00 AM',
        maxValue: '6:00 PM',
        increment: 30
    }]
  });
});
```

как и в нашем последнем примере этот простой FormLayout содержит один предмет, в нашем случае TimeField. Нажатие на триггер и у нас есть опции определенные нашим кодом как время между 9 утра и 6 вечера с инкрементов в полчаса. Другие опции настройки позволят определять несколько принимаемых форматов времени, а также применить собственную валидацию. Простая валидация дается по умолчанию.



NumberField
NumberField не выглядит каким-то особенным, это просто поле 'text'. Его особенность заключается в том что он позволяет встроить в наше приложение защиту от ложных команд. Клиентская валидация данных жизненно важно для удостоверения в целостности данных для того чтобы отправлять значения на сервер, позволяя нам предотвращать ошибки от удаленных вызовов процедур. Такие компоненты как этот упрощают нам подобные процессы. NumberField автоматически фильтрует нажатия клавиш, разрешая ввод только чисел, что дает нам возможность указать может ли оно использовать отрицательные или плавающие значение, или даже как много десятчиных мест следует принимать.

CheckboxGroups и RadioGroups
Ext JS 2.2 привнес две новые формы управления : CheckboxGroup и RadioGroup. Эти управления позволяют нам "группировать" наборы чекбоксов или радио-кнопок, давая им собственное форматирования и специальные, на уровне группы, возможности валидации. Например, представим, что у нас есть форма с двумя чекбоксами из которых пользователю предстоит выбрать свой пол. Разместив их внутри CheckboxGroup, мы можем применить нашу валидацию к группе, таким образом по меньшей мере одна опция будет выбрана при отправке формы. Мы можем применять и другие сложные правила валидации, в зависимости от случая. Для более подробной информации читайте API Ext

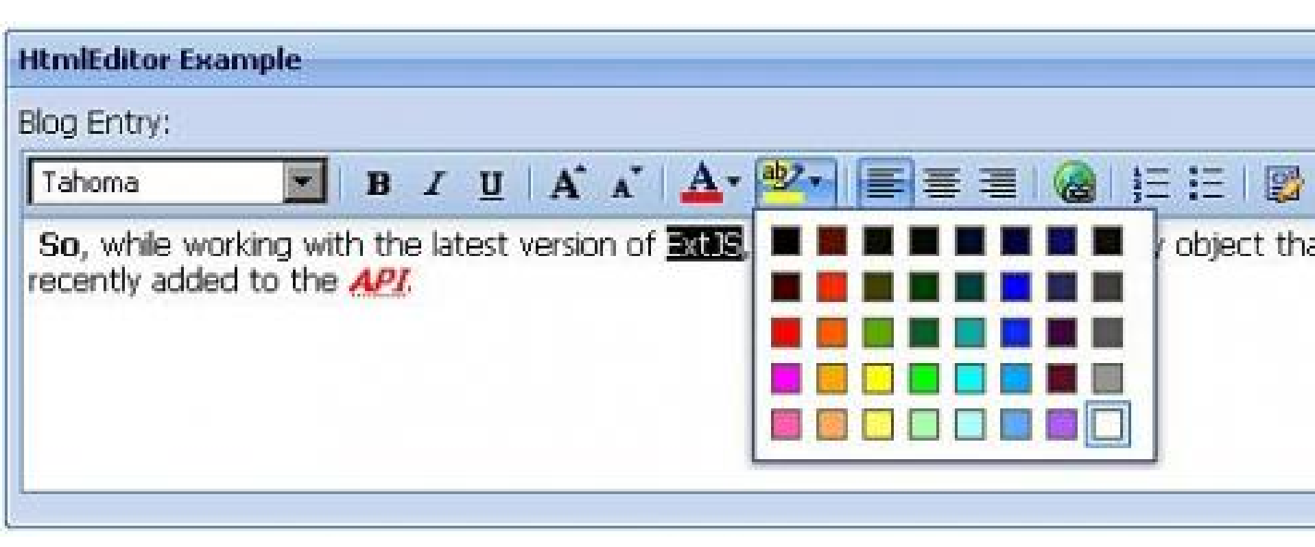
HtmlEditor
В **Ext JS** включен неплохой, ограниченный, WYSIWYG HtmlEditor, чтобы размещаться прямо в ваших формах:
Example 3: chapter14_03.js

Исходный код примера:

Исходный код примера:

```
Ext.onReady(function(){
  var formPanel = new Ext.form.FormPanel({
    title:'HtmlEditor Example',
    applyTo:'chap14_ex03',
    layout:'form',
    labelAlign:'top',
    width:600,
    autoHeight:true,
    frame:true,
    items:[{
      xtype:'htmleditor',
      id:'bio',
      fieldLabel:'Blog Entry',
      height:200,
      anchor:'98%'
    }]
  });
});
```

Здесь мы запихнули HtmlEditor в FormLayout с целью быстрой демонстрации. HtmlEditor применяется к простым текстовым областям, почти так же как и другие редакторы WYSIWYG, например FCKEditor или TinyMCE. Мы можем включать или выключать различные кнопки меню относящиеся к форматированию так же как и вызывать различные методы или применять слушатели. Все как в остальных компонентах **Ext JS**.



So, while working with the latest version of Ext JS, … object that was recently added to the API.

У нас все выглядит точно так же как и должно быть в **Ext JS**, но при этом дает пользователю возможность изменять введенные им данные изнутри нашего приложения. На изображении выше. обратите внимание что HtmlEditor использует встроенный в **Ext JS** компонент ColorPalette для выбора цвета.

Форматирование данных
Мы все знаем, что мы не всегда получаем данные в том формате, в котором хотели бы его отобразить. **Ext JS** предоставляет различные компоненты специально работающие с такими случаями. В первую очередь это объект Format в пакете Ext.util, который дает широкий выбор функций для всего, от создания формата американского доллара и до нескольких методов для разбора кода **HTML** из строк. Библиотека **Ext JS** также расширяет несколько родных для **JavaScript** объектов и дает нам огромный выбор дополнительных методов для управления ими. Особенно были расширены объекты String, Number и Date. Теперь вы можете избавиться от пустых мест, заставить цифры принимать минимальные и максимальные значения, и даже создавать объекты Date из различного количества опций формата.

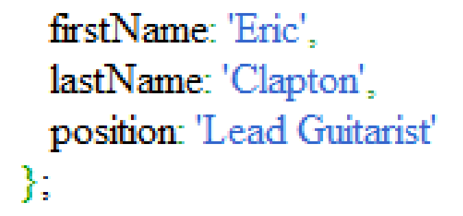
Простое форматирование строк
Объект String был расширен для предоставления нескольких форматирующих опций, включая метод format(). Этот простой метод позволяет вернуть отформатированный текст с первым параметром являющимся строкой возврата, а все остальные параметры являются частями строки, используя основные выражения привязки, где фигурные скобки заключают ссылку переменной. Типичные выражения привязки в английском, и используемые **Ext JS** будут содержать ссылки на переменные с точечным обозначением и окруженные фигурными скобками (это, {this.firstName}), но в этом примере мы будем привязывать значения к аргументам, которые будут следить за порядком массива:
Example 4: chapter14_04.js

Исходный код примера:

Исходный код примера:

```
Ext.onReady(function(){
  var cls = "Band";
  var member = {
    firstName: 'Eric',
    lastName: 'Clapton',
    position: 'Lead Guitarist'
  };
  var pnl = new Ext.Panel({
    applyTo: 'chap14_ex04',
    width: 200,
    height: 100,
    bodyStyle: 'padding:5px',
    title: 'Band Member',
    html: String.format('<div class=\'{0}\'>{2}, {1}: {3}</div>',
          cls, member.firstName, member.lastName, member.position)
  });
});
```

Этот маленький блок кода отображает простой элемент Panel, вызывающий функцию форматирования для построения **HTML** панели. Метод Format(), а точнее его аргументы, применяются к различным частям строки которую мы строим.



Clapton, Eric: Lead Guitarist

Форматирование датFormatting dates
Объект Date object также был расширен для проставления дополнительного форматирования иди возможностей парсинга. **API** дает законченные слушатели для идентификаторов частей, которые могут быть использованы при парсинге строки в объект или переводе Date в нужный формат:
Example 5: chapter14_05.js

Исходный код примера:

Исходный код примера:

```
Ext.onReady(function(){
  var cls = "Band";
  var member = {
    firstName: 'Eric',
    lastName: 'Clapton',
    position: 'Lead Guitarist',
    birthDate: new Date('03/30/1945')
  };
  var pnl = new Ext.Panel({
    applyTo: 'chap14_ex05',
    width: 200,
    height: 100,
    bodyStyle: 'padding:5px',
    title: 'Band Member',
    html: String.format('<div class=\'{0}\'>{2}, {1}: {3}<br />DOB:
{4}</div>', cls, member.firstName, member.lastName, member.position,
member.birthDate.format('F j, Y'))
  });
});
```

Мы расширили наш предыдущий пример добавили день рождения Эрика. мы создали новый объект Date в качестве части объекта Member и затем использовали метод format() объекта Date для изменения отображаемого формата данных.

- 1
- 2
- 3
- следующая ›
- последняя »

- English

Голос