[Обучение Ext JS] :: Часть 3. Создание форм в Ext

Фев 18 2010**Встроенные подтверждения—vtypes**
Другой, более сложный способ подтверждения это vtype. Он может использоваться для подтверждения и ограничения ввода пользователя, и вывода сообщения об ошибке. Это сработает любым сценарием, так как применяются обычные выражения для выполнения всей грубой работы.
Несколько полезных встроенных vTypes:

- email
- url
- alpha
- alphanum

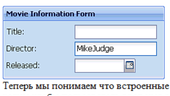
Эти встроенные *vtypes* являются самыми простыми и в основном используются как отправная точка для создания собственных *vtypes*. Пример использования *alpha vtype* совместно с *QuickTips* с сообщении об ошибке:

Исходный код примера:

Исходный код примера:

```
Ext.onReady(function(){
    var movie_form = new Ext.FormPanel({
        url: 'movie-form-submit.php',
        renderTo: document.body,
        frame: true,
        title: 'Movie Information Form',
        width: 250,
        items: [{
            xtype: 'textfield',
            fieldLabel: 'Title',
            name: 'title',
            allowBlank: false
        },{
            xtype: 'textfield',
            fieldLabel: 'Director',
            name: 'director',
            vtype: 'alpha'
        },{
            xtype: 'datefield',
            fieldLabel: 'Released',
            name: 'released',
            disabledDays: [1,2,3,4,5]
        }]
    });
});
```

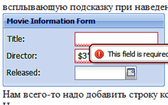
Все что мы сделали, это добавили vtype в поле оператора. Он будет проверять чтобы все вводимые знаки были буквенными.

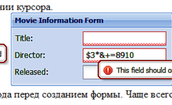


Теперь мы понимаем что встроенные vtypes очень просты. Vtype alpha вводит ограничение ввода на все, кроме букв. В нашем случае мы хотим чтобы пользователь ввел имя режиссера, которое содержит только буквы. Заглавная буква в начале имени, возможно, сделает его красивее.

ВАЖНО: *Поищите на форуме **Ext**, и вы, возможно, найдете vType созданный другим пользователем и который может быть как раз тем, который вам нужен (или стать отправной точкой для ваших собственных экспериментов)*

Стили для вывода ошибок
Формы, установленные по умолчанию, обладают очень мягким окном сообщения об ошибке. Оно отображает ее извилистой красной линией под полем формы. Очень похоже на то, как Microsoft Word подчеркивает неправильно написанное слово. А ведь у нас есть другие способы сообщить об ошибке! Но для этого нам надо сказать **Ext JS**, что он должен использовать их.
Самый распространенный способ вывода ошибке - в виде облака. Это не только убирает стандартное подчеркивание, но и добавляет всплывающую подсказку при наведении курсора.





Нам всего-то надо добавить строку кода перед созданием формы. Чаще всего это строка заключенная в функцию *OnReady*.
Например:

```
Ext.onReady(function(){
    Ext.QuickTips.init();
    // our form here
});
```

Это все, что требуется для того, чтобы отображать ошибку в симпатичном облачке.

Пользовательское подтверждение—создайте свой собственный vtype
Если вы, похожи на меня, то стандартные выражения порой могут вас поставить в тупик. Поэтому я всегда пытаюсь найти что-то близкое к тому что мне надо, а потом подогнать под мою задачу (а не начинать с чистого листа).
Для создания собственного vtype нам понадобиться ввести определения vtype. У каждого определения есть значения, маска, текст ошибки и функция используемая при проверке:

- *xxxVal*: Это обычное выражение
- *xxxMask*: Маскирующее ограничение ввода пользователя
- *xxxText*: Выводимый текст ошибки

Как только мы выясним какие обычные выражения нам необходимы, можно приступать к созданию нашего собственного vType. Рассмотрим на примере подтверждения правильности ввода имени нашего режиссера. Обычное выражение рассматривает пару строк alpha, разделенных пробелом и каждая начинается с заглавной буквы. Звучит неплохо, да?

Исходный код примера:

Исходный код примера:

```
Ext.form.VTypes['nameVal']  = /^[A-Z][A-Za-z\-]+
                              [A-Z][A-Za-z\-]+$/;
Ext.form.VTypes['nameMask'] = /[A-Za-z\- ]/;
Ext.form.VTypes['nameText'] = 'In-valid Director Name.';
Ext.form.VTypes['name']     = function(v){
  return Ext.form.VTypes['nameVal'].test(v);
}
```

Возможно не все понятно при взгляде на все это в сборе. Поэтому рассмотрим еще и по частям. Мы начинаем с обычного выражения, которое проверяет правильность введенных в форму данных:

```
Ext.form.VTypes['nameVal'] = /^([A-Z]{1})([A-Za-z\-]+
                             ([A-Z]{1})([A-Za-z\-]+/;
```

Затем, мы добавляем маскировку, которая определяет какие знаки могут быть вписаны в нашу форму. Это тоже обычное выражение:

```
Ext.form.VTypes['nameMask'] = /[A-Za-z]/;
```

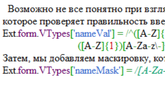
Затем прописываем текст, выводимый в сообщении об ошибке:

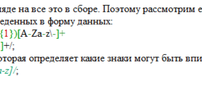
```
Ext.form.VTypes['nameText'] = 'In-valid Director Name.';
```

И наконец, часть которая удерживает все это вместе и заставляет работать. Функция проверяющая значение нашего поля:

```
Ext.form.VTypes['name'] = function(v){
  return Ext.form.VTypes['nameVal'].test(v);
}
```

Соберите все это вместе и у вас получится свой vtype, при том без больших усилий. И так можно делать снова и снова





Маскировка, или "не нажимайте эту клавишу!"
Маскировка используется для того, чтобы заставить поле принимать только буквы или цифры (или то и другое но без спецсимволов). Возможности безграничны, потому что вы сами выбираете какие клавиши разрешить, а какие запретить. Этот пример маскировки показывает как можно использовать только заглавные буквы:
maskRe: /[A-Z]/
Вместо использования настроек маскировки, создайте *vType* для того, чтобы завершить маскировку. Если вдруг того потребуют условия, и придется изменять что-либо, то вы всегда сможете легко найти это место.

Кнопки с зависимой фиксацией и чекбоксы
Кнопки и чекбоксы это необходимое зло. Они неуклюжи, и с ними трудно работать. Я использую их только тогда, когда уже ничего больше не остается. Но давайте добавим их в наш код, просто для того, чтобы сказать, что мы это сделали.

Это не просто кнопка, а кнопка с зависимой фиксацией
Давайте сначала добавим одну из кнопок в нашу форму:

Исходный код примера:

Исходный код примера:

```
{
  xtype: 'radio',
  fieldLabel: 'Filmed In',
  name: 'filmed_in',
  boxLabel: 'Color'
},{
  xtype: 'radio',
  hideLabel: false,
  labelSeparator: '',
  name: 'filmed_in',
  boxLabel: 'Black & White'
}
```

Эти кнопки будут работать так же как и их HTML аналоги. Дайте им всем одно и то же имя, они будут работать вместе. Я также люблю прятать ярлыки к кнопкам выставляя значение hideLabel на true и оставляя значение labelSeperator пустым. Так форма будет выглядеть чище.

Поставьте галочку
Иногда нам необходимо использовать чекбоксы для булевых значений — что-то вроде рубильника *вкл/выкл*.

```
{
  xtype: 'checkbox',
  fieldLabel: 'Bad Movie',
  name: 'bad_movie'
}
```