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

Фев 18 2010

В этой главе мы научимся создавать формы **Ext**, которые похожи на формы **HTML**, но которые не ограничены по применению и обладают удобным интерфейсом. Мы используем слегка отличающиеся поля форм для создания формы, которая будет подтверждать достоверность и асинхронно инициализировать выполнение. Затем, мы создадим выпадающие меню (ComboBox) с использованием базы данных и добавим еще много чего, например сложные поля подтверждения и маскировку. А затем, мы закончим все это парой серьезных тем, которые значительно улучшат наши формы.

Цели этой главы:
- Создать форму которая использует **AJAX**
- Подтвердить достоверность полей информации и создать собственное подтверждение
- Загружать данные из базы

Основные части формы

С формами **Ext** возможности безграничны. Подтверждения, сообщения об ошибках и ограничения по значениям, все это создается при помощи простых настроек опций. Расширение настроек формы под ваши требования выполняется очень легко, и позже мы обязательно к этому вернемся. Я перечислю некоторые основные компоненты с которыми вы должны быть знакомы::
- *Ext.form.FormPanel:* Группирует поля в панель. Во многом похоже на действие тега FORM в обычной форме HTML
- *Ext.form.Field:* Главный оператор создания формы и взаимодействия с ней. Можно сравнить с тегом INPUT

Наша первая форма

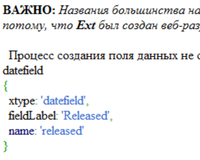
Для начала создадим форму с разнообразными полями, сборщиком информации, сообщениями об ошибке и использующей **AJAX** — на первый раз самую простую. Для этого примера наши поля будут созданы с использованием настройки объекта, вместо компонента *Ext.form.Field* . Этот метод прекрасно работает, не требует больших затрат по времени и поможет исполнять код быстрее. Начальная страница **HTML**, вроде той которую мы использовали в нашем прошлом примере может использоваться как отправная точка. Необходимо прописать стандартные файлы библиотеки **Ext**, и как и все что мы создаем в ней, наш код должен быть заключен в функцию *onReady*.

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'
        },{
            xtype: 'textfield',
            fieldLabel: 'Director',
            name: 'director'
        },{
            xtype: 'datefield',
            fieldLabel: 'Released',
            name: 'released'
        }]
    });
});
```

Когда мы запускаем код в браузере, у нас в итоге должна появиться вот такая панель формы:



Милая ворма. А как она работает?

FormPanel очень похожа на форму **HTML**. Она ведет себя как хранилище для наших полей формы. У формы есть url-конфигурация, таким образом она знает куда отправлять полученную информацию. У нее так же есть настройка renderTo, которая определяет когда форма выводится на странице. Элемент настроек предметов очень важен, поскольку содержит все наши поля форм.
И представляет собой массив полей. У каждого элемента поля есть xtype, который определяет какой тип компонента **Ext** будет использован: текст, дата или число. Компонентом может быть даже сетка или другой тип компонента **Ext**.

Поля форм

Теперь мы знаем, что каждый тип поля определяется при помощи xtype. Но откуда этот xtypes берется и как часто встречается? Xtype, этовсего лишь ссылка на какой-либо определенный компонент **Ext**, таким образом xtype 'textfield' представляет собой то же что и Ext.form.TextField. Некоторые доступные нам xtypes:
- textfield
- timefield
- numberfield
- datefield
- combo
- textarea

Поскольку они все являются компонентами **Ext**, мы с легкостью можем использовать сетку, панель инструментов или кнопки (да вообще все что угодно). Главное в компонентах **Ext** это то, что они взаимозаменяемы и у них общие основные функции. Поэтому любой сценарий может быть выполнен библиотекой **Ext**.
Наши базовые поля содержат примерно такие установки::

```
{
  xtype: 'textfield',
  fieldLabel: 'Title',
  name: 'title'
}
```

Конечно у нас есть *xtype* который определяет какой тип поля используется (в нашем случае *textfield*). *fieldLabel* , это текстовый ярлык отображаемый слева от поля, хотя его можно перенастроить и он будет отображаться сверху или справа. Настройка *name* то же самое что используется в **HTML** и будет использовать переменное имя, когда отсылает данные формы серверу.

ВАЖНО: *Названия большинства настроек компонентов **Ext** совпадает с их аналогами в **HTML**. А все потому, что **Ext** был создан веб-разработчиками для веб-разработчиков.*

Процесс создания поля данных не сильно отличается от создания поля текста. Просто измените *xtype* на datefield

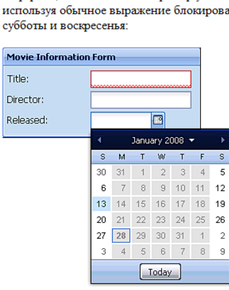
```
{
  xtype: 'datefield',
  fieldLabel: 'Released',
  name: 'released'
}
```

Подтверждение

Некоторые примеры нашего кода могут иметь подтверждения, которые выдают пользователю сообщение с ошибкой, в случае если он делает что-то не то. Давайте добавим подтверждение в нашу первую форму. Один из наиболее часто используемых типов это проверка на то ввел ли пользователь какое-либо значение или нет. Мы используем такое для нашего поля заголовка. Другими словами придадим полю следующий вид:

```
{
  xtype: 'textfield',
  fieldLabel: 'Title',
  name: 'title',
  allowBlank: false
}
```

Установка *allowBlank* и добавление *false* (по умолчанию стоит *true*) достаточно простое задание. У большинства создаваемых нами форм будет много требуемых полей.
У каждого типа поля Ext есть собственный набор специальных подтверждений которые зависят от типа информации этого поля. Например у поля *date* есть возможность блокировать некоторые дни недели, или используя обычное выражение блокировать определенные даты. Пример блокирующий все дни кроме субботы и воскресенья:



В этом примере все дни, кроме выходных, заблокированы. Помните что неделя начинается с 0 (воскресенья) и заканчивается на 6 (суббота). Когда мы используем другие типы полей, мы используем другие типы подтверждения, например поля ввода цифр, которые можно ограничить по количеству знаков. Стандартные настройки подтверждений для каждого типа поля могут быть найдены по ссылке на API.