Главная › Блоги › Lertion's блог

# [Обучение Ext JS] :: Часть 9. Окна и диалоги

Мар 04 2010

В былые дни сети, пользователь традиционных серверных систем, потратил бы время вгрызаясь список с данными и интерфейс на формах. Выбрать предмет из списка, перейти к форме с информацией, вернуться и повторить. Проблема в том, что мы говорим о тысячах записей и большого количества информации в формах. Существует вероятность того, что в нашем примере потребительского заказа понадобятся малые формы для того, чтобы показать всю информацию. И каждый раз, когда мы переходим на другой экран, мы обновляем страницу и получаем все данные. Снова.

С этим все нормально. Так работает сеть. Но в случае обработки данных, когда пользователи весь день используют сайт, стоит задуматься над оптимизацией скорости, на которой появляются экраны и обновляются данные. Ключевую роль, привнося постраничную навигацию основанную на **AJAX** и сортировку GridViews, в этом играет таблица.
Теперь, мы обратим наше внимание на другую часть головоломки — **Ext JS** Window и поддержка диалогов. Эти классы позволяют разработчикам выдавать пользователям любой вид информации, не заставляя их при этом переходить на другой экран. При помощи всплывающего поверх основного содержимого окна или диалога, они могут отобразить всю интересующую пользователя информацию в виде таблиц, дерева, картинок и текста. Мы так же можем использовать их упрощенную форму для вывода предупреждений или для захвата пользовательских данных.

**Начиная диалог**
В этой главе мы поговорим о главном классе Ext.Window, и подклассе Ext.MessageBox, у которых есть обширные приложения в нашем улучшенном пользовательском интерфейсе. В то время как Window создана для того чтобы быть гибким и многоцелевым элементом, MessageBox является более специализированным решением. Оно используется таким же образом что и стандартное всплывающее окно **JavaScript**, хотя у него и есть много поведенческих и презентационных опций.
Ext.Window другой полностью сформировавшийся Ext.Container, дает хороший набор основных настроек которые знакомы по другим частям инфраструктуры **Ext JS**. Nтак же это дает понять какие типы интерфейсов можно создать в окне, и настроить внутреннюю компоновку контейнера при помощи большого количества опций.
Мы также рассмотрим несколько дополнительных классов которые помогут нам управлять несколькими окнами: Ext.WindowGroup и Ext.WindowManager. В расширенных приложениях можно использовать больше чем одно окно для вывода глубинной информации, или мы можем позволить пользователю просматривать больше одной записи в отдельном окне. Группы окон помогают этим приложениям, давая возможность работать с большим количеством окон.
Несмотря на тот факт, что диалоговые окна построены на Ext.Window, мы собираемся сначала обратится к ним. Диалоги, из Ext.Window много полезных функций, превращая его в подрезанную версию суперкласса.

**Диалоги**
Как было сказано ранее, диалоги похожи на быстрые и предупреждающие функции, доступные для реализации **Javascript** в большинстве браузеров. Несмотря на то, что внешний вид этих всплывающих окон подчиняется браузеру, и операционной системе, а их поведение ограничено только самыми общими случаями, эти ограничения не распространяются на диалоги **Ext JS** dialogs.

**ВАЖНО:** *Хотя класс диалога на самом деле Ext.MessageBox, в **Ext JS** существует сокращенная версия - Ext.Msg. Он может сделать код более сжатым и лаконичным, но окончательный выбор за вами. По функциональности они равны.*

Давайте посмотрим на уже сделанное всплывающее окно, которое доступно в **Ext JS**.

**С полки**
Мы говорили о том, как **Ext** заменяет предупреждения **JavaScript**, поэтому сначала давайте посмотрим на это:
`Ext.Msg.alert('Hey!', 'Something happened.');`
Первое, что здесь заметно, то что у Msg.alert два параметра, в то время как у стандартного предупреждения только один. Первый позволяет уточнить название предупреждающего диалогового окна, а второе уточняет текст в его теле. Этот код приводит вот к такому окну сообщения:

Something happened.

Как вы видите, по функциям оно делает то же самое, что и обычное предупреждение, но со знакомым нам внешним видом **Ext JS**. Мы также можем передать чуть больше информации используя строку названия. Вывод Msg.alert не прекращает выполнение скрипта, как это происходит с нормальным предупреждением. Позже мы покажем как использовать обратные вызовы для того чтобы в случае необходимости копировать функцию остановки.

**ВАЖНО:** *В одну единицу времени можно пользоваться только одним Ext.MessageBox. Если вы попытаетесь вызвать два окна в одно и то же время, первое заменит второе. В некоторых случаях вам может понадобиться проверить существующий диалог, в случае если вы невнимательно переписали его сообщение.*

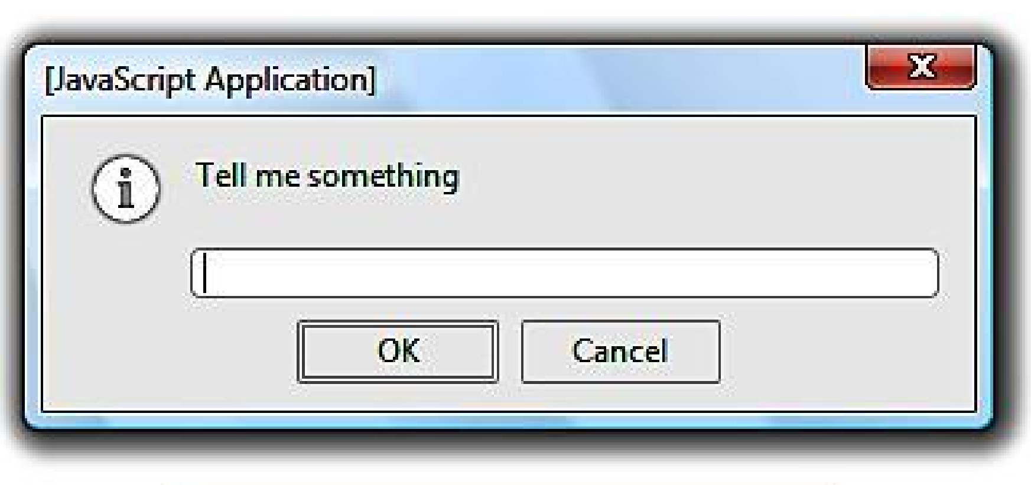
Давайте посмотрим на другую замену, Ext.Msg.prompt. Она позволяет захватывать строку текста совсем как обычная подсказка JavaScript. Но вместо простого возвращения значений она дает больше опций. Использование метода для уже знакомой нам вещи:

Исходный код примера:

## Исходный код примера:

```
var data = prompt('Tell me something');
Ext.Msg.prompt('Hey!', 'Tell me something', function(btn, text){
  if (btn == 'ok'){
    var data = text;
  }
}, this, false, '');
```

И снова, Msg.prompt позволяет передать название как первый агумент, а тело теста является вторым аргументом. Третий аргумент - функция обратного вызова которыая вызывается при нажатии на любую из кнопок (OK или Cancel) окна. У обратного вызова есть два аргумента: кнопка, которую нажали, и текст, что был предоставлен пользователем. Можно использовать это, как показано в примере кода.



Tell me something

Tell me something

Обратите внимание на другие три опции, которые идут после функции обратного вызова. Ими соответственно являются область, многострочность и начальное значение. Многострочность интересна тем, что принимая логическое значение она позволяет более гибкий захват текста чем стандартное приглашение.

**Подтверждение**
Мы окончательно заменим окно сообщения на всплывающее окно с подтверждением, которое даст пользователю возможность выбирать между подтверждением действия и отказом от него. Теперь код должен быть нам знаком:

Исходный код примера:

## Исходный код примера:

```
Ext.Msg.confirm('Hey!', 'Is this ok?', function(btn, text){
   if (btn == 'yes'){
     // go ahead and do more stuff
   } else {
     // abort, abort!
   }
});
```

И снова мы используем название, текст тела и аргументы обратного вызова которые уже видели в the Msg.prompt. Интересная разница между этим и стандартным подтверждением заключается в том, что там, где стандартное предлагает пользователю выбрать между OK и Cancel, **Ext JS** дает на выбор Yes и No. Это бесспорно лучший выбор, особенно если сообщении используется вопрос.

**Все это приятно продолжается**
Четвертый стандартный messagebox содержащийся в **Ext JS** не просто замена обычному всплывающему окну **JavaScript**. Это диалог хода выполнения. Ext.Msg.progress не может быть использован отдельно как другие типы, и не требует пользовательского ввода. На самом деле, если вы активируете его таким образом:
`Ext.Msg.progress('Hey!', 'We\'re waiting...', 'progressing');`
То вам придется подождать, потому что у вас появится модальное окно диалога, которое не будет меняться. первые два аргумента являются названием и текстом тела. В то время как третий это текст, который появляется на полосе состояния.
Таким образом если мы не хотим застрять на вечном прогрессбаре, то как заставить ее двигаться? Как раз для этого и существует метод Ext.Msg.updateProgress. Вот пример иллюстрирующий его применениe:

Исходный код примера:

## Исходный код примера:

```
var count = 0;
var interval = window.setInterval(function()
  count = count + 0.04;
  Ext.Msg.updateProgress(count);
  if(count >= 1) {
    window.clearInterval(interval);
    Ext.Msg.hide();
  }
}, 100);
```

Это очень притянутый за уши пример, который вызывает updateProgress каждые 100 миллисекунд через таймер, и продвигает полосу используя каждый раз переменную count. Первым аргументом updateProgress является значение между нулем и еденицей, где ноль, это начало, а еденица - конец. Второй позволяет изменять текст прогрессбара, а третий изменяет текст тела. изменение текста может оказаться полезным, в случае если вы хотите установить с пользователем дополнительную связь. Даже если это всего-лишь проценты.
Вернемся к нашему примеру кода. Обратите внимания, что вы можете вызвать Ext.Msg.hide для того, чтобы скрыть диалог хода выполнения — updateProgress за вас этого не сделает, даже если вы установите текущее значение прогресса больше единицы.
Эти четыре типа диалоговых окон являются основой поддержки диалога **Ext JS**, но мы вполне можем приспособить их и для других случаев.

**Развиваем усвоенное**
Все четыре только что освоенные нами являются переходом к пятому: Ext.Msg.show. Этот метод берет настраиваемый объект как единственный аргумент, и опция настройки в объекте позволяет создавать messagebox поддерживающий все особенности доступные через методы переходов. Простейшим примером такого кода является:

```
Ext.Msg.show({
   msg: 'AWESOME.'
});
```

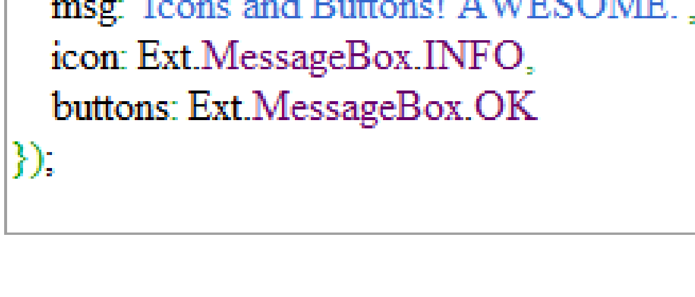
Это самая близкая копия предупреждения **JavaScript** кторую воспроизводит **Ext JS**. Но она не обладает достаточным функционалом. Вот эта лучше:

Исходный код примера:

## Исходный код примера:

```
Ext.Msg.show({
   title: 'Hey!',
   msg: 'Icons and Buttons! AWESOME.',
   icon: Ext.MessageBox.INFO,
   buttons: Ext.MessageBox.OK
});
```

Мы вернули название и кнопки. Но кроме того теперь прибавляется иконка.



Icons and Buttons! AWESOME.

Способы настройки кнопок и иконок довольно интересны: отправить одну из постоянных предоставляемых **Ext JS**, и получите уже настреонную кнопку или класс **CSS** который отображает иконку. Вот список констант для иконок:

- Ext.Msg.ERROR
- Ext.Msg.INFO
- Ext.Msg.QUESTION
- Ext.Msg.WARNING

А эти для кнопок:

- Ext.Msg.CANCEL
- Ext.Msg.OK
- Ext.Msg.OKCANCEL
- Ext.Msg.YESNO
- Ext.Msg.YESNOCANCEL

Эти готовые опции дают неплохой кусок гибкости, когда приходит пора отображать messagebox, но мы пойдем дальше. Как уже было сказано, константы иконок, это просто строки представляющие названия классов **CSS**. Например, Ext.Msg.QUESTION выводит строку ext-mb-question. Эти вписывается в стиль **Ext JS** и предоставляет стили для формы вопроса. Логический вывод, который из этого следует, заключается в том, что мы можем поставить нашу собственную строку на место этих постоянных, что позволит полностью менять настройки зон иконок.
Константы кнопок менее гибкие, и содержат буквенные объекты, уточняющие каким именно образом должны отображаться диалоговые кнопки. Например Ext.Msg.YESNOCANCEL содержит следующее:

```
{ cancel:true, yes: true, no:true }
```

Это говорит о том что необходимо включить все кнопки yes, cancel, и no. Можно селективно отключать кнопки, но нельзя изменить их порядок или добавить новые. А значит это дает несколько ограничивает способы их использования.

**ВАЖНО:** *В дополнение к приему констант кнопок Ext.Msg, опция метода show, также принимает стандартные объекты настройки кнопок **Ext JS**.*

Тем не менее мы можем пригладить окно диалога другими способами. Возможно снабдить Ext.Msg.show опциями высоты и ширины, для того чтобы ограничить размеры всплывающего окна. Это может оказаться полезно в случаях, при которых надо вывести длинное сообщение, и вы не хотите чтобы оно растянулось в огромную строку на весь экран.
Объект настройки show также позволяет использовать опцию cls для уточнения класса **CSS** чтобы применить его к контейнеру диалога. Разработчик может использовать это для того чтобы выбрать любой дочерний объект контейнера использующего собственные правила **CSS**, для того, чтобы иметь потенциал нескольких диалогов с совершенно разными внешними видами. Вы хотите удивить пользователя ярко-розовым всплывающим окном? Эта опция настроек позволит вам осуществить это.

- 1
- 2
- 3
- 4
- следующая ›
- последняя »