[Обучение Ext JS] :: Часть 11. Drag-and-Drop

Мар 16 2010**Зоны контроля**
Мы узнали как выцепить отдельный перетаскиваемый предмет, но чаще всего надо перетащить несколько элементов готовых к перемещениям. Решением здесь является использование Ext.dd.DragZone и Ext.dd.DropZone для разрешения перемещения нескольких узлов. Мы скопируем простой список приложений, который позволит пользователю перемещать элементы из одного списка в другой. Тем не менее этот функционал будет серьезно ограничен, позволяя нам сосредоточиться на важных темах. Сначала настроим немного базового **HTML**:

Исходный код примера:

Исходный код примера:

```
<h1>Today</h1>
<ul id="today">
  <li>Shopping</li>
  <li>Haircut</li>
</ul>
<h1>Tomorrow</h1>
<ul id="tmrw">
  <li>Wash car</li>
</ul>
```

У нас есть два списка, один с id today а второй с id tmrw. Наш пользователь сможет перемещать элементы между этими двумя списками, имитируя реорганизацию заданий на несколько дней. Наш первый фрагмент JavaScript будет использовать Ext.dd.DragZone для установки этих списков в качестве контейнеров для перетаскиваемых предметов Помните, что вашему коду **JS** надо заходить в вызов Ext.onReady.

```
new Ext.dd.DragZone('today');
new Ext.dd.DragZone('tmrw');
```

Это выглядит довольно прямолинейно, но есть один важный момент. Испытание кода в этом месте обнаружит то, что на самом деле он ничего не включает. Вместе с настройкой DragZone, нам надо эксплицитно сказать которые дочерние узлы в DragZones будут переносимы.

ВАЖНО: *Зачем так надо? В сложных приложениях может быть очень много вложенных дочерних узлов. Таким образом, одновременное обозначение их как переносимых может вызвать проблемы в работе и неожиданное поведение. Вместо этого мы можем вручную настроить это, избегая возможного вреда.*

Для того чтобы элементы нашего списка могли быть перенесены, надо зарегистрировать их при помощи:

```
var drags = document.getElementsByTagName('li');
for(var i =0; i< drags.length; i++){
  Ext.dd.Registry.register(drags[i]);
}
```

Это довольно наивно, простая регистрация элементов на странице, но она дает понять что требуется. Если вы проверите код теперь, то увидите что все элементы стали переносимы. Теперь надо перейти к созданию места назначения:

```
var cfg = { onNodeDrop: drop }
new Ext.dd.DropZone('tmrw', cfg);
new Ext.dd.DropZone('today', cfg);
```

Мы создали объект cfg, который содержит общие настройки данных для обоих случаев Ext.dd.DropZone. Метод onNodeDrop для DropZone автоматически вызывается DropZone.notifyDrop когда он понимает, что вы отпустили зарегистрированный узел. В этом случае мы указываем на функцию, вызвавшую это, что в таком случае мы перепишем существующее и пустое применение onNodeDrop. Новое выглядит таким образом:

Исходный код примера:

Исходный код примера:

```
// dropNodeData-drop node data object
// source-drag zone which contained the dragged element
// event-the drag drop event
// dragNodeData-drag source data object
// this-destination drop zone
function drop(dropNodeData, source, event, dragNodeData) {
var dragged = source.dragData.ddel;
  var sourceContainer = source.el.dom;
  var destinationContainer = this.getEl();
  sourceContainer.removeChild(dragged);
  destinationContainer.appendChild(dragged);
  return true;
}
```

Мы используем аргументы, отправленные к этой функции для удержания трех элементов: перетаскиваемого предмета, списка из которого он перетаскивается и список куда его перетаскивают. С этой информацией мы можем удалить предмет из списка источника и переместить в другой список. Затем вернуть значение true чтобы предотвратить возвращение заместителя на прежнее место.

Изменяем наши списки

Этим примером кода мы на практике показали как использовать функционал **Ext JS** drag-and-drop. Конечно это очень простое приложение, но оно хорошее.

Позже мы рассмотрим многие из аспектов поддержки инфраструктуры **Ext JS** drag-and-drop с помощью очень малого количества настроек. И выясним почему важно знание того, как работают классы основы.

Регестрируем заинтересованность

Мы уже обсудили то, как должны быть зарегистрированы дочерние узлы DragZone в Ext.dd.Registry прежде чем их можно будет переносить. Этот дополнительный шаг может быть болезненным, и чуть позже мы это несомненно заметим, но сейчас давайте сосредоточимся на преимуществах получаемых когда узел добавлен в Ext.dd.Registry.
В предыдущих примерах мы использовали метод Ext.dd.Registry.register с единственным параметром: элемент который надо зарегистрировать. Тем не менее регистрация занимает секунду, опциональный параметр позволяющий добавить объект метаданных в перетаскиваемый предмет. Эти метаданные могут использоваться как вам только будет удобно, но самое вероятное - отправить информацию о "совместимости" к событию сброса. Представьте, что вы создаете приложение, которое позволяет собирать "виртуальную внешность" перетаскиванием предметов одежды. Можно использовать метаданные для того, чтобы убедиться что только, к примеру, могут сочетаться вещи только из летней коллекции. Осуществляется это, включением ряда информации в данные переноса. Существует много разных приложений использующих метаданные, но поскольку они просто третьесторонние объекты, как их использовать решать вам.
Хотя и существуют служебные протоколы, связанные с использованием Ext.dd.Registry, мы так же показали что у него есть важные преимущества, которые, в зависимости от задач стоящих перед вашим приложением могут перевесить проблемы записи дополнительного регистрационного кода.

Экстремальный drag-and-drop

Одна их великих вещей в **Ext JS** drag-and-drop заключается в том, что можно использовать множество предоставляемых компонентов, без необходимости лезть в дебри и узнавать как это все работает. Мы собираемся рассмотреть использование drag-and-drop для создания детально проработанной компоновки при помощи Ext.DataView и Ext.FormPanel. Таким образом мы можем показать интеграцию drag-and-drop и наиболее часто используемые компоненты. Большая часть из того что мы собираемся рассмотреть может показаться знакомой по предыдущим примерам, но приложение будет отличаться.
Мы будем работать с очень простым:

```
<div id="people"></div>
<div id="detail"></div>
```

<div> будет содержать наш DataView, список группы людей изымаемый из хранилища. <div> detail будет содержать форму куда мы сможем перетаскивать предметы.

Перетаскивание DataView

Первым шагом будет создание нашего Ext.DataView, и разрешение на перенос узлов. Код для инициализации этого выглядит таким образом:

Исходный код примера:

Исходный код примера:

```
var personView = new Ext.DataView({
  tpl: '<tpl for=".">' +
       '<div class="person">' +
         '<h1>{name}</h1>' +
         '<div><img src="img/{image}" alt="{name}" /></div>' +
        '</div>' +
      '</tpl>',
   itemSelector: 'div.person',
   store: personStore
});
personView.render('people');
```

Вы заметите, что применение personStore не показано, но ему необходимо иметь предметы ID, name, image, city, и country для того, чтобы поддерживать то что мы собираемся сделать. Обратите внимание что в нашем шаблоне мы показываем содержащий имя человека заголовок и его изображение.
основываясь на нащем опыте со списком предметов мы можем установить DragZone для контейнера the DataView и зарегистрировать элементы <div> используя Ext.dd.Registry. Попробовав сделать так, вы поймете почему это не для такого типа приложений. Оно просто не работает. Узлы person не могут быть перенесены. Причина заключается в Ext.dd.Registry. Детям DragZone надо зарегистрироваться в Ext.dd.Registry, прежде чем они смогут быть перенесены. нам надо удоствоериться что каждый возможный узел был зарегистрирован. Это определенно не очень практично, и потому нам нужно какое-то другое решение.

Разберемся с переносом данных

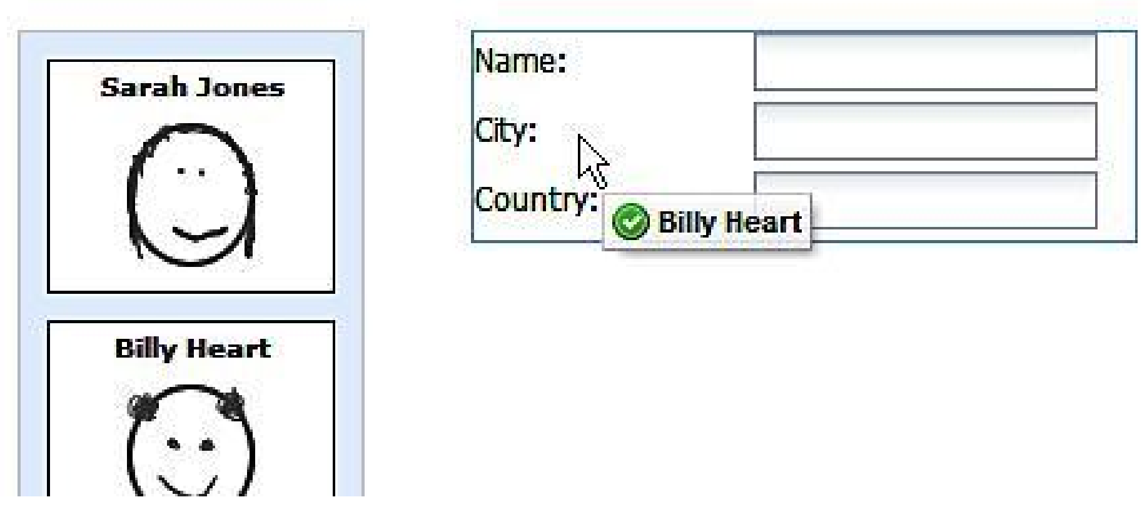
нам надо решить эту проблему на немного более высоком уровне, а не заниматься по отдельности каждым дочерним узлом. Для достижения нашей цели, мы перепишем поведение Ext.dd.DragZone.
Метод getDragData класса DragZone вызывается всякий раз, когда мышь совершает действие а пределах контейнера DragZone. Это значит, что мы можем использовать слушатели для начала события переноса вне зависимости от тоо какой дочерний узел был задействован. В наших обстоятельствах мы будем изучать цель события и переходить через **DOM** до тех пор, пока не найдем тот узел, который действительно хотим использовать. Таким образом код для DragZone будет выглядеть так:

Исходный код примера:

Исходный код примера:

```
new Ext.dd.DragZone(personView.getEl(), {
   getDragData : function(e) {
      var container = e.getTarget('div.person', 5, true);
      return {
         ddel : container
      }
   }
});
```

как упоминалось ранее, мы получаем узел контейнера person от события, и мы используем его для наполнения свойства объекта ddel которое вернулось от getDragData. Ранее в этой главе мы разбирались с отпусканием узла в DropZone, и рассматривали свойство dragData.ddel для того, чтобы найти отпускаемый узел. Это одни и те же данные, но с другой точки зрения — вот так она наполняется.



Теперь у нас есть рабочая DataView, но есть еще пара интересных добавлений, которые могут показаться стоящими. Давайте рассмотрим их.

Заместители и метаданные

Когда мы указываем свойство ddel в getDragData, мы по существу, наполняем заместитель, который будет следовать за курсором мышки по мере перетаскивания узла. Это полезное соображение. Код такой:

```
return {
  ddel : container.down('h1').dom
}
```

Теперь, вместо того, чтоб наш заместитель появлялся как целый узел, он отображается как признак h1 в пределах нашего узла. Мы говорили раньше о заместителях и упоминали, что перенос обрезанной версией предмета может быть очень полезен. И как это можно осуществить используя DragZones.
У нас также есть опция отправки дополнительных данных в перенос, используя другие свойства dragData. Фактически, мы добавляем объект, который будет доступен когда предмет будет отпущен. Вот пример кода того, что мы собираемся сделать:

```
return {
  ddel : container.down('h1').dom,
  record : personView.getRecord(container.dom)
}
```

Мы можем передать запись связанную с предметом по которому щелкнули мышкой в нашу операцию переноса. Это может очень пригодиться, когда мы дойдем до наполнения нашей FormPanel.

Отпускание в деталях

Есть и обратная сторона медали — обработка действия отпускания, которая включается когда мы отпускаем переносимый из нашей DataView. Концепты, лежащие за этим, будут знакомы по предыдущим примерам. Нам надо создать DropTarget из FormPanel, а затем переписать его метод notifyDrop, для указания, что произойдет когда случится действие отпускания.
Давайте посмотрим на код для нашей FormPanel:

Исходный код примера:

Исходный код примера:

```
var detailForm = new Ext.form.FormPanel({
   width: 250,
   height: 80,
   defaultType: 'textfield',
   items: [
      { fieldLabel: 'Name', name:'name' },
      { fieldLabel: 'City', name:'city' },
      { fieldLabel: 'Country', name:'country' }
   ]
});
```

Все это довольно стандартно, но обратите внимание на названия полей формы, совпадающие с названиями полей в записях наших данных которые были использованы для наполнения переносимого узла. Теперь мы можем настроить DropTarget:

Исходный код примера:

Исходный код примера:

```
new Ext.dd.DropTarget(detailForm.body.dom, {
   notifyDrop : function(source, e, data){
      var record = source.dragData.record;
      detailForm.getForm().loadRecord(record);
      return true;
   }
});
```

Обратите внимание, что мы можем захватить всю запись прямо из dragData по источнику переноса и загрузить ее в форму. Это сильная сторона, размещать любые данные в dragData, и здесь показано как можно наполнить форму с минимумом кода.

Группы Drag-drop

Мы начали экскурсию по Ext.dd с того что рассмотрели то, как можно переносить индивидуальные узлы. Теперь мы опишем методы, которые позволят нам переносить сразу несколько узлов и как они могут быть отпущены в контейнер с дочерними узлами.
Когда вы работаете с большим количеством переносимых элементов, существует средство сделать так, чтобы узлы вели себя так, как вы от них ждете: группы drag-drop. DragSource, DragZone, DropTarget, и DropZone принимают опцию настройки ddGroup которые в свою очередь принимают идентификатор строки, показывающий какой группе вы хотите назначить ваш случай.
Но что действительно делает эта опция и что нового даст она вашему приложению? Ну, говоря без обиняков, это значит что случай класса drag-drop будет способен взаимодействовать с другими случаями drag-drop в той же самой группе, но это объяснение довольно сухое. Давайте рассмотрим как это используется.

- « первая
- ‹ предыдущая
- 1
- 2
- 3
- 4
- следующая ›
- последняя »

- English

Голос