Главная › Блоги › Lertion's блог

[Обучение Ext JS] :: Часть 8. Ext JS растет на деревьях

Фев 28 2010**XML**
Деревом **XML** изначально не поддерживается. Тем не менее, чтобы это стало возможным, можно воспользоваться поддержкой данных **Ext JS**. Использование **JSON** снимает часть проблем, связанных с этим, хотя некоторые приложения могут использовать **XML** для передачи данных. Думаю, стоит обсудить некоторые основные подходы.
Для того, чтобы достать данные мы можем использовать Ext.data.HttpProxy, Но для того чтобы прочитать **XML** нам понадобиться немного потрудиться:

Исходный код примера:

Исходный код примера:

```
var xmltree = new Ext.tree.TreePanel({el: 'treeContainer'});
var proxy = new Ext.data.HttpProxy({url:
                'http://localhost:81/ext/treexml.php'});
proxy.load(null, {
  read: function(xmlDocument) {
    parseXmlAndCreateNodes(xmlDocument);
  }
}, function(){ xmltree.render(); });
```

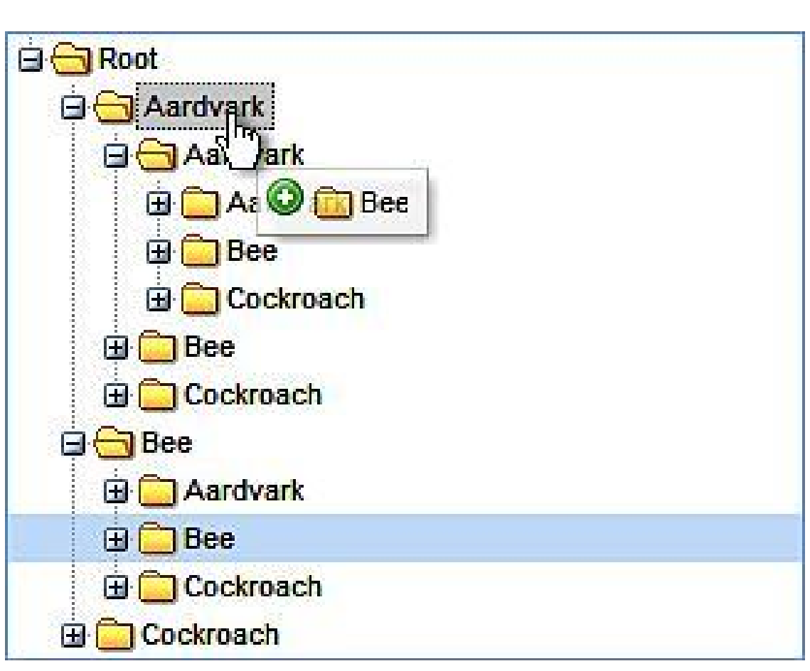
Мы создаем новую TreePanel и HttpProxy, а затем указывает что когда прокси загружается, мы хотим, чтобы Ext.data.Reader обработал входящие **XML** данные. После этого, мы говорим считывателю отправить **XML** к parseXmlAndCreateNodes. В этой функции вы создадите корневой TreeNode и детей, основываясь на полученных данных, которому довольно просто дать понять что HttpProxy осведомлен об **XML** и отправит вам настоящий документ **XML**, а не его строки.
JavaScript полностью способен обработать данные **XML**, хотя для вас, возможно, гораздо удобнее получасть как траверс **DOM** документа **XHTML**. Исследуя и читая документ **XML** вы можете построить иерархию TreeNode, заключение атрибутов **XML** как дополнительных данных для каждого узла и использование textnodes как текстовых ярлыков. Поскольку, таким образом, у вас есть доступ к сырым узлам **XML**, у вас есть полный контроль над результатами древовидной структуры и
TreeNodes составляющий ее.

Ухаживаем за деревьями
Теперь мы собираемся обсудить главные особенности которые вы, возможно, захотите прикрутить к своему деревус целью повышения полености. Drag-and-drop, сортировка и изменение узлов - это то, что превращает TreePanel из хитрого способа отображения данных в отличный способ управления ими.

Drag-and-drop
Ext JS позаботится обо всем UIdrag-and-drop когда вы используете TreePanel. Просто добавьте enableDD: true в ваши настройки и вы сможете перестроить узлы и добавить их в папки.

ВАЖНО: *The TreePanel заботится не только о своих узлах. Если у вас на странице больше чем одна TreePanel, то вы можете переносить ветви и листья между ними.*

Но это только половин дела. Когда вы обновите страницу, все ваши перестройки исчезнут. Это потому, что TreePanel не знает как вы хотите сохранить ваши изменения и чтобы научить ее, нам надо подцепить кое-какие события.



Событие beforemovenode, как раз то, что нам нужно. Оно срабатывает после того как мы отпускаем копку мыши, но до того как UI TreePanel обновится и отразит это. Мы, вполне возможно, захотим добавит код, похожий тот что приведен ниже, для того, чтобы сказать серверу о событии перемещения узла:

Исходный код примера:

Исходный код примера:

```
tree.on('beforemovenode', function(tree, node,
                              oldParent, newParent, index) {
   Ext.Ajax.request({
      url: 'http://localhost/node-move.php',
      params: {
        nodeid: node.id,
        newparentid: newParent.id,
        oldparentid: oldParent.id,
        dropindex: index
      }
   });
});
```

Дорабатывая наш предыдущий код мы добавляем новый обработчик событий для события beforemovenode. Функция обработчика вызывается несколькими полезными аргументами:

1. tree: TreePanel вызвавшая событие
2. node: TreeNode который переместили
3. oldParent: Предыдущий родитель перемещенного узла
4. newParent: Новый родитель перемещенного узла
5. index: Численный индекс места, куда был перемещен узел

Мы можем использовать эти аргументы для формирования параметров запроса **AJAX** к серверу. Поскольку вы можете вытащить довольно много информации, вам нужно текущее состояние дерева. А серверный скрипт может выполнить любое действие необходимое для этого.
В некоторых случаях это может включать отметку действия переноса. Если в некоторых случаях,логика, которую вы включили в обработчик beforemovenode не работает, отмените свои изменения. Если вы не выполняете запрос **AJAX**, то это довольно просто — верните значение false в конце обработчика, и действие будет отменено. Впрочем, для **AJAX** это более сложно, поскольку **XMLHttpRequest** происходит асинхронно и обработчик событий перейдет к действию по умолчанию, которое позволяет перенос.
В этих обстоятельствах вам надо быть уверенными в том, что обеспечите отказ обработчика запроса **AJAX**, и отправите достаточно информации, для того, чтобы вернуть дерево в предыдущее состояние. Для этого мы воспользуемся beforemovenode, который передает достаточно данных для того, чтобы позаботится об этих событиях отказа.

Сортировка
Мы можем очень гибко отсортировать узлы в TreePanel, используя TreeSorter. И опять, взяв за основу наш предыдущий код, мы можем создать вот такой TreeSorter:

```
new Ext.tree.TreeSorter(tree, {
  folderSort: true,
  dir: “asc"
});
```

Поскольку TreeSorter берет на себя ответственность за пару умолчаний — в особенности то, что ваши узлы листьев помечены листом с соответствующим названием, и что ваши ярлыки в тексте с соответствующим названием — мы можем с легкость выполнить сортировку по алфавиту. Параметр dir говорит TreeSorter отсортировать либо по восходящей (при значении asc), либо по нисходящей (desc), праметр folderSort означает, что следует отсортировать узлы листьев, расположенные в папках. Другими словами, всю иерархию дерева.
Если у вас есть данные, не являющиеся простым текстом, вы можете указать свой способ сортировки при помощи опции настройки sortType. SortType берет функицю как свое значение и эта функция будет вызвана одним аргументом: TreeNode.
Цель функции sortType заключается в том, чтобы бросить пользовательское свойство TreeNode— вероятно то, что вы отправили с сервера и это то, что вам нужно. Сконвертировать в формат, который **Ext JS** может отсортировать. Другими словами, в один их стандартных типов такие как целое число, строка и данные.
Эта особенность может быть полезна в тех случаях, когда данные отправляются к дереву в формате, которые не способствует нормальному поиску. Данные, сгенерированные сервером, могут служить нескольким целям, и в то же время не всегда подходить для каких-то определенных целей. Например, нам необходимо сконвертировать данные в стандартный формат —из американского ММ/ДД/ГГ в ГГГГММДД, который подходит для сортировки — или, мы можем захотеть убрать посторонние знаки из денежного значения, чтобы переделать его в десятичное число.

```
sortType: function(node) {
  return node.attributes.creationDate
}
```

В вышеприведенном примере мы вернули некоторые пользовательские данные из узла. И поскольку это значение является допустимым для **JavaScript**, **Ext JS** сможет его отсортировать. Это простая демонстрация того, как может быть использована опция sortType для того, чтобы позволить TreeSorter работать с любым видом серверных данных.

Редактирование
Существует много вариантов развития событий, при которых редактирование значений узлов может оказаться полезным. Когда просматриваете иерархию категоризованных продуктов вы, возможно, захотите переименовать или категорию или продукт. Без перехода на другой экран. Мы можем сделать эту простую особенность используя класс Ext.tree.TreeEditor class:

```
var editor = new Ext.tree.TreeEditor(tree);
```

Значения выставленные в TreeEditor по умолчанию означают, что это даст вашим узлам дерева редактор TextField при двойном щелчке на ярлыке. Тем не менее как и в случае с функцией drag-and-drop включение не значит то, что изменения будут сразу же сохраняться. Нам необходимо обработать событие запускающееся после завершения редактирования узла:

```
editor.on('beforecomplete', function(editor, newValue, originalValue)
{
  // Possible Ajax call?
});
```

Обработчик события beforecomplete вызывается тремя аргументами:

1. editor: Поле редактора необходимое для изменения узла
2. newValue: Введенное значение
3. originalValue: Значение до изменений

Тем не менее важно заметить, что параметры редактора не обычный Ext.form.Field. Он расширены несколькими дополнительными свойствами, наиболее полезное из которых, editNode, ссылка на отредактированный узел. Это позволяет вам получить вам ID узла, что может быть полезно при серверном запросе на синхронизацию значений в базе данных.

Событие beforemovenode, как и в случае с TreePanel, позволяет отменить редактирование узла, возвращая значение в конец обработчика. Запросу **AJAX** придется дать обработчик отказа для того, чтобы вручную восстановить измененные значения.
Это был быстрый взгляд на то, как создать простой встроенный редактор. Так же есть способы использования этого класса для создания более сложных особенностей. Конструктор TreeEditor может взять до двух опциональных параметров, поверх одного обязательного показанного в примере чуть выше. Эти параметры: объект конфигурации поля и объект конфигурации для TreeEditor. Конфигурация поля может быть двух видов: объект настройки поля применяемый к стандартному редактору или уже созданный пример другой формы поля. В последнем случае вы сможете похожим образом добавить NumberField, DateField или другую Ext.form.Field.
Второй параметр позволяет настраивать TreeEditor, и скорее больше подходит для тонкой настройки, чем для какого-либо расширенного функционала. Например, мы можем использовать cancelOnEsc чтобы разрешить пользователю отменять любое изменение при нажатии клавиши Escape, или использовать ignoreNoChange чтобы избежать завершения события если значение не было изменено после редактирования.

Стрижка и подрезка
Есть еще несколько трюков поддерживаемые TreePanel, которые помогают в создании хороших приложений. Изменяя модели выбора, фильтровки узла и контекстные меню наиболее часто используемые возможности в самых различных решениях. Давайте рассмотрим их.

Модели выбора
В нашем предыдущем примере кода мы перетаскивали и редактировали TreeNodes чтобы добиться их изменения немедленно. Но узды можео выбрать и для последующей обработки. TreePanel по умолчанию использует одиночную модель выбора. В нашем предыдшем коде мы уже сделали все что нужно для включения выбора узлов. Как и со многими другими аспектами, простой выбор ничего не делает. Нам нужно разместить в нем несколько свойств, предназначенных для управления выбором.
Отличным примером этого будет выбор узла и автоматическое наполнение панели информации подробными данными об этом самом узле. Возможно у вас есть дерево продуктов, и нажатие на узел отобразит цену и наличие на складе. Чтобы это произошло мы используем событие selectionchange. Используя наш прошлый код как отправную точку мы добавим в него следующее:

```
tree.selModel.on('selectionchange', function(selModel, node) {
   var price = node.attributes.price;
});
```

Второй аргумент узла, отправляемый к событию selectionchange позволяет с легкостью взять любой пользовательский атрибут в данных узла.
Что если мы захотим позволить выбор нескольких узлов? как можно такое сделать и как в таком случае обработать событие selectionchange? Можно использовать Ext.tree.MultiSelectionModel во время создания нашей TreePanel:

Исходный код примера:

Исходный код примера:

```
var tree = new Ext.tree.TreePanel({
      renderTo:'treeContainer',
      loader: treeLoader,
      root: rootNode,
      selModel: new Ext.tree.MultiSelectionModel()
  });
```

Настройка тоже простая. Хотя обработка события selectionchange очень похоже на стоящую по умолчанию модель выбора существуют важные различия. Второй аргумент обработчика события будет массивом узлов.

Модели выбора не просто показывают способы получения информации. Они также позволяют управлять текущим выбором. Например метод MultiSelectionModel.clearSelections() полезен для чистки реестра, после того как того как закончится обработка события с несколькими узлами. У DefaultSelectionModel существуют способы (selectNext и selectPrevious) для переходов по дереву, вверх и вниз по иерархии узла.

Собираемся с контекстными меню
Мы уже рассмотрели много функционала предоставляемого TreePanel, так что давайте закрепим это на практике. Добавление контекстного меню, которое будет появляться при щелчке правой кнопкой мыши на TreeNode для **Ext JS** задание самое обычное. Впрочем, это может оказаться очень полезным способом добавления ссылок в ваш интерфейс. За основу мы возьмем код из предыдущего раздела. Сначала, давайте создадим меню, а затем встроим его в TreePanel:

Исходный код примера:

Исходный код примера:

```
var contextMenu = new Ext.menu.Menu({
  items: [
    { text: 'Delete', handler: deleteHandler },
    { text: 'Sort', handler: sortHandler }
   ]
});
tree.on('contextmenu', treeContextHandler);
```

TreePanel предоставляет событие contextmenu которое запускается, когда пользователь щелкает правой кнопкой мышки на узле. Обратите внимания что наши слушатели не анонимные функции (как это было в прошлом примере), вместо этого они были разделены для упрощения чтения.
Сначала treeContextHandler обрабатывающий событие contextmenu:

```
function treeContextHandler(node) {
  node.select();
  contextMenu.show(node.ui.getAnchor());
}
```

Обработчик вызывается аргументом узла, так что нам выбрать узел для того, чтобы позже можно было действовать. Затем мы вызываем всплывающее меню используя способ с одним параметром, который скажет ему относительно чего ему следует выравняться. В нашем случае это текст TreeNode на котором мы щелкнули.

Обрабатываем меню
У нас есть два объекта контекстного меню — Delete и Sort. Давайте посмотрим на обработчик Delete:

```
function deleteHandler() {
  tree.getSelectionModel().getSelectedNode().remove();
}
```

Мы просто выбираем узел в treeContextHandler, и вызываем его способ удаления. Это удалит и узел и всех его детей из TreePanel. Обратите внимания, что мы не разбираемся с сохранением этих изменений на сервере, но это надо сделать. У TreePanel есть событие даления и можно использовать обработчик для наших целей.
Обработчик для объекта меню Sort:

Исходный код примера:

Исходный код примера:

```
function sortHandler() {
  tree.getSelectionModel().getSelectedNode().sort(
    function (leftNode, rightNode) {
      return (leftNode.text.toUpperCase() < rightNode.text.
toUpperCase() ? 1 : -1);
    }
  );
}
```

И снова мы используем модель выбора чтобы получить выбранный узел. **Ext JS** дает способ сортировки TreeNode который берет функцию как первый параметр. Эта функция вызывается двумя аргументами: двумя узлами которые нужно сравнить. В этом примере мы сортируем текст узла в нисходящем порядке. Но вы можете отсортировать узел так, как вы хотите.

ВАЖНО: *Вы можете использовать этот метод сортировки в сочетании с TreeSorter без проблем. Это возможно потому, что TreeSorter отслеживает только события beforechildrenrendered, append, insert, и textchange. Любые другие изменения не окажут влияния.*

Действие Delete в контекстном меню полностью удалит выбранный узел из TreePanel, в то время как действие Sort упорядочит его детей в соответствии с их ярлыками.

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