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

Фев 28 2010

С иерархической структурой данных большинство разработчиков знакомы очень близко. Структура корень-ветвь-лист используется во многих интерфейсах, от представления файлов и папок в Windows Explorer до классического гениалогическоо дерева со всеми внуками правнуками и их родителями. Пакет Ext.tree позволяет разработчикам использовать эти структуры данных всего лишь несколькими строчками кода. И дает неплохую возможность решать трудные случаи при помощи набора простых опций настроек.

Хотя набор иконок прилагающийся к **Ext JS** по умолчанию показывает узлы дерева как файлы и папки, он не ограничен концептом файловой системы. Иконки и текст элементов (или узлов вашего дерева), можно изменить основываясь на динамическом или статическом типе данных, которые используются для их наполнения — и не спрашивая пользовательский код. Как насчет экрана безопасности, отображающего группы с разрешениями где в каждой группе указано число пользователей и приложена иконка фотографии, или галерея фотографий, или превью изображения в иконке. С помощью класса древовидной схемы все это становится возможным.

Насаждение на будущее

В конце-концов дерево **Ext JS** не заботится об отображаемых данных. Оно достаточно гибкое и может справиться с любым вариантом развития событий. Данные могут загружаться как предварительно, так и логическими пакетами. Это может стать важной особенностью, в случае если у вас очень много информации подлежащей загрузке. Вы можете редактировать данные прямо в дереве, изменяя ярлыки и местоположение элементов. Или же вы можете изменить внешний вид всего дерева, или отдельно взятого узла. Все это даст в итоге неплохой опыт.

Дерево **Ext JS** построено поверх модели Component, лежащей в основе всего оформления **Ext JS**. Это значит то, что разработчики находятся в выгодном положении, используя знакомую систему, пользователи получают согласованный и встроенный использования интерфейса, а вы можете быть уверены в том, что ваше дерево будет безупречно работать с остальной частью вашего приложения.

Из маленького семечка...

В этой главе мы рассмотрим как построить дерево начав с первых принципов содержащих минимум кода. Мы также обсудим уникальную структуру данных, используемую для наполнения дерева, и способ при котором умное использование этих данных позволяет использовать важные опции настроек. Дерево **Ext JS** по умолчанию поддерживает такие особенности как сортировку и drag-and-drop. Так что, их мы тоже обсудим. Но если вам необходимо по настоящему особенное дерево, мы обсудим способы при помощи которых можно заменить или расширить опции настроек, методы и события.

Само дерево создается через класс Ext.tree.TreePanel, который в свою очередь, содержит маркер классов Ext.tree.TreeNodes. Эти два класса являются основой для поддержки дерева **Ext JS**, и потому будут главной темой для обсуждения на протяжении всей этой главы. Впрочем, существуют и другие, не менее важные классы которые мы с вами рассмотрим. Вот полный список из Ext.tree package:

AsyncTreeNode	Разрешает асинхронную загрузку детей TreeNode
DefaultSelectionModel	Стандартный выбор одного элемента для TreePanel
MultiSelectionModel	Разрешает выбор нескольких узлов
RootTreeNodeUI	Специализированная TreeNode для корня TreePanel
TreeDragZone	Предоставляет поддержку для перетаскивания TreeNode
TreeDropZone	Предоставляет поддержку для отпускания TreeNode
TreeEditor	Разрешает редактировать ярлыки узлов
TreeEditor	Поддержка фильтров для узлов детей TreePanel
TreeLoader	Наполнение TreePanel из указанной URL
TreeNode	Основной класс, представляющий узел в TreePanel
TreeNodeUI	Предоставляет основы интерфейса для TreeNode
TreePanel	Древовидное представление данных — главный класс дерева
TreeSorter	Древовидное представление данных — главный класс дерева

Ой! к счастью, вам не придется использовать их все сразу. TreeNode и TreePanel дадут вам основу. Остальные классы специально заточены под большую функциональность. Мы рассмотрим каждый из них когда придет время. Обсудим как они используются и приведем несколько примеров их использования.

Наш первый побег

К этому моменту вы возможно думаете о разнообразных возможностях дерева **Ext JS** и с нетерпением ждете возможности приступить. Не смотря на тот факт, что классы Ext.tree наиболее богаты на полезности из всех доступных, вы по прежнему можете запустить их парой строк кода.

В последующем примере мы удостоверимся, что у вас есть пустой реестр страницы **HTML**, со всеми включенными зависимостями **Ext JS**. Большая часть кода будет использовать то, что мы уже рассматривали, и потому работать будет только с нашим небольшим объемом. Помните об этом, когда будете рассматривать их по отдельности.

Самой лучшей практикой будет поместить **JavaScript** в отдельный файл и свернуть его в обращение Ext.onReady call. Впрочем, можно придерживаться собственного стиля написания кода.

Подготовка места

Сначала нам надо создать на нашей **HTML** странице элемент содержащий <div>. Мы будем рендерить TreePanel в этот контейнер. таким образом нам надо настроить ее под такой размер, каким должно быть наше дерево:

```
<div id="treecontainer" style="height:300px; width:200px;"></div>
```

JavaScript для дерева может быть разбит на три части. Сначала нам надо уточнить способ которым он будет наполняться. Ext.tree. класс TreeLoader предоставляет этот функционал, и вот самый простой способ это сделать:

```
var treeLoader = new Ext.tree.TreeLoader({
   dataUrl:'http://localhost/samplejson.php'
});
```

Параметр настроек dataUrl указывает местоположение скрипта, который будет обеспечивать работу **JavaScript Object Notation (JSON)**, отвечающего за наполнение нашего дерева. Я не собираюсь вдаваться в детали и расписывать структуру **JSON**. Оставим это на потом..

Каждое дерево требует корневой узел, действующий как пра-прадедушка для всех его потомков. Для создания такого узла мы используем класс Ext.tree.AsyncTreeNode:

```
var rootNode = new Ext.tree.AsyncTreeNode({
   text: 'Root'
});
```

Причина по которой мы используем AsyncTreeNode, а не TreeNode, заключается в том, что мы, извлекая узлы с сервера хотим чтобы дети были наполнены ветвь за ветвью а не все разом. Это самый обычный сценарий для дерева.

ВАЖНО: *AsyncTreeNode использует **AJAX** на ходу, затем, чтобы пользователь не ждал загрузки данных или рендеринга первых узлов.*

Наконец, мы создаем собственно дерево, используя класс Ext.tree.TreePanel:

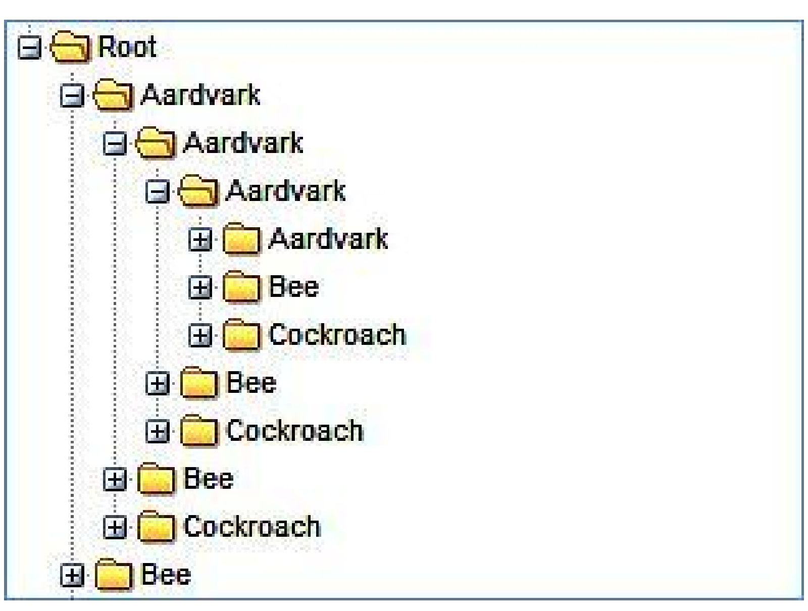
```
var tree = new Ext.tree.TreePanel({
   renderTo:'treecontainer',
   loader: treeLoader,
   root: rootNode
});
```

Это просто вопрос передачи TreeLoader как опций настроек, и использование настроек renderTo для уточнения того, что мы хотим чтобы TreePanel была отрендерена в наш элемент treeContainer.

И снова, помните, что вы должны свернуть весь этот код в запрос Ext.onReady, для того чтобы удостовериться в доступности **DOM** при выполнении кода.

Дерево не может расти без данных

Мы уже знаем что требуется всего лишь одиннадцать строк кода для создания класса дерева используя интерфейс **Ext JS**. Вот пример конечного продукта:



Полагаю, выглядит не очень, но для одиннадцати строк это обладает отличной функциональностью. Мы получили вполне приличный вид с удаленной асинхронной загрузкой удаленных дочерних узлов. Честно говоря это не так просто как кажется потому что мы едва задели одну важную часть построения дерева — данные.

JSON

Стандартный TreeLoader поддерживает **JSON** в особом формате — массиве определения узлов. Пример использования:

Исходный код примера:

Исходный код примера:

```
[
  { id: '1', text: 'No Children', leaf: true },
  { id: '2', text: 'Has Children',
    children: [{
      id: '3',
      text: 'Youngster',
      leaf: true
    }]
  }
]
```

Свойство text является меткой узла, как это является в дереве. Свойство id используется для уникольной идентификации каждого узла и будет использоваться для определения какой узел выбран или развернут. Применение id сделает вашу жизнь намного проще, есливы используете некоторые расширенные возможности TreePanel,которые мы рассмотрим чуть позже. Свойство children опционально. Про свойство leaf можно считать отмечающим узел папкой или файлом. В дереве листовые узлы не будут развертываемы и у них не будет иконки "плюс", которые присущ папкам.

Пара слов о ID

По умолчанию TreeNodes присваивается автоматически сгенерированный ID, и значит, что свойство настройки ID опционально. Сгенерированный ID является строкой текста вида ynode-xx, где xx заменяется номером. ID могут быть полезны для нахождения узла, на котоырй вы ссылались. Как бы то ни было, вполне может статься так, что вы захотите назначить ID самостоятельно. В любое время когда вы разворачиваете узел с детьми и запускаете асинхронную загрузку данных с сервера, вашему серверному скрипту необходимо знать на какой именно узел вы нажали для того, чтобы отослать вам детей. При самостоятельном выставление узлам ID будет проще сопоставить их с выполняемыми действиями при работе с сервером.

Дополнительные данные

Хотя свойства id, text, and leaf являются наиболее используемыми, то как он наполняют **JSON** не единственный способ. На самом деле любое свойство настроек TreeNode может быть инициализировано **JSON**, что будет довольно полезно, когда мы начнем рассматривать другие возможности дерева. Вы также сможете включить специализированные данные. Возможно ваши узлы это продукты, и вы хотите установить на них цену. Любое свойство, которое не распознается как опция настройки TreeNode по прежнему будет включена для последующего доступа в свойство TreeNode.attributes.

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