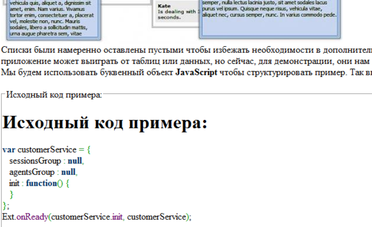
[Обучение Ext JS] :: Часть 9. Окна и диалоги

Мар 04 2010**пример нескольких окон**

Мы создадим очень простую имитацию приложения работающего с покупателями. Наш пользователь будет диспетчером живой чат-системы, где покупатели будут спрашивать вопросы онлайн и получать на них ответы. Диспетчер будет брать некоторые из вопросов, но он так же отслеживает состояние других агентов, взаимодействующих с системой. Нам будет интересен только аспект окон, так что нам придется заполнить их большим количеством пустышек.
Вот скриншот простого приложения разрабатываного чтобы поддерживать такую систему:



Списки были намеренно оставлены пустыми чтобы избежать необходимости в дополнительном коде **JavaScript** — подобное приложение может выиграть от таблиц или данных, но сейчас, для демонстрации, они нам не нужны.
Мы будем использовать буквенный объект **JavaScript** чтобы структурировать пример. Так выглядит оболочка:

Исходный код примера:

Исходный код примера:

```
var customerService = {
  sessionsGroup : null,
  agentsGroup : null,
  init : function() {
  }
};
Ext.onReady(customerService.init, customerService);
```

У нас есть пара локальных переменных которые будут держать наши группы окон, и пустые вызываемые функции initТеперь рассмотрев скелет, мы необходимо посмотреть на разметку, придающую силу нашему приложению:

Исходный код примера:

Исходный код примера:

```
<div id="mySessions">
  <h2>My Sessions
    <button id="hideSessions">Hide All</button>
    <button id="tileSessions">Tile</button>
  </h2>
  <div id="s-p1">
    <h3>Bill</h3>
    <p>Started at 12:31pm</p>
    <div class="content"></div>
  </div>
</div>
<div id="agents">
  <h2>Agents
    <button id="hideAgents">Hide All</button>
    <button id="tileAgents">Tile</button>
  </h2>
  <div id="a-h1">
    <h3>Heidi</h3>
    <p>Is dealing with 3 sessions...</p>
    <div class="content"></div>
  </div>
</div>
```

Это **HTML** идущий прямо в тело нашего документа. У нас есть два главных контейнера — mySessions и agents — который в свою очередь содержит теги h2с парой кнопок и набором тегов div. Например, в mySessions, у нас есть div #s-p1, содержащий h3, параграф и другой div с классом содержания. #s-p1 div один из тех, которые могли появится в пределах mySessions, с этими div, названными #s-xxx, где xxx является уникальным идентификатором сессии.
Внутри агентов у нас сходная структура, но с несколько другим ID — например агент divs называются #a-xxx. Различные ID в **HTML** очень важны в **JavaScript**, поскольку используются для оплова обработчиков событий, как мы увидим через секунду. Давайте посмотрим на полную функцию init.

Исходный код примера:

Исходный код примера:

```
init : function() {
  var s = Ext.select;
  var g = Ext.get;
  this.sessionGroup = new Ext.WindowGroup();
  this.agentsGroup = new Ext.WindowGroup();
  s('#mySessions div').on('click', this.showSession, this);
  s('#agents div').on('click', this.showAgent, this);
}
```

Первые две линии кода в этом блоке просто ссылке для того чтобы подчистить код. Мы сортируем отсылки на функции **Ext** в виде переменной со сходным названием. Следующие две строки создают группы окон. Одну, которая будет содержать информацию о сессии вторая для агентов окон информации. Следующие две линии придают смысл обсуждаемому ранее **HTML**.
Мы используем вызовы для Ext.select и добавляем слушатели события во все div в #mySessions — обрабатываемой функцией showSession — и в #agents — обрабатываемый showAgent. Третий аргумент функции удостоверяется что мы остаемся в пределах границ когда вызываются обработчики. Рассмотрим функцию showSession:

Исходный код примера:

Исходный код примера:

```
showSession : function(e){
  var target = e.getTarget('div', 5, true);
  var sessionId = target.dom.id + '-win';
  var win = this.sessionsGroup.get(sessionId);
  if(!win) {
    win = new Ext.Window({
      manager: this.sessionsGroup,
      id: sessionId,
      width: 200,
      height: 200,
      resizable: false,
      closable: false,
      title: target.down('h3').dom.innerHTML,
      html: target.down('.content').dom.innerHTML
    });
  }
  win.show();
  win.alignTo(target);
}
```

Первая строка удостоверяется, что наша целевая переменная содержит разделение сеанса, на которое нажали, независимо от того, было ли это фактически одним из его дочерних элементов. Затем мы сохраняем уникальный sessionId, беря идентификатор целевого отделения сеанса и прилагая к нему -win. Наконец, мы проверяем, есть ли уже у группы окна сеанса справочная информация по окну которое мы собираемся создать, поискав его, используя уникальный sessionId.
Если окно не существует,то нам надо его создать. Большинство опций настроек нам уже знакомы, но обратите внимание на экстенциональное назначение sessionsGroup как на диспетчер группы для этого окна. Мы также используем sessionId как ID для окна, не полагаясь на автоматически создаваемый ID используемый по умолчанию. С названием и опциями html мы используем Ext.Element устройствами обхода **DOM** с целью назначить разделителям сессии текст h3 и соответственно элементам div.content.
Теперь мы уверены что переменная win содержит достоверную ссылку на окно, и можем продвигаться дальше. После того как мы все сделали, мы хотим выравнять разделитель на который нажали и отобразить его, так что вызывается функция окна alignTo. Это остановка по требованию для нашей функции showSession. Функция showAgent почти идентична, но ссылается на agentsGroup а не sessionsGroup. Совместно с этими двумя функциями убеждаются что наши окна сессии и агент успешно появляются, таким образом сейчас время повнимательнее посмотреть как можно использовать WindowGroups для управления.

WindowGroups служба работы с покупателями

Мы можем добавить две линии в нашу функцию init которая использует группы окна для выполнения расчистки рабочего места нашей службы:

```
g('hideSessions').on('click', this.sessionsGroup.hideAll);
g('hideAgents').on('click', this.agentsGroup.hideAll);
```

Можно добавить ловушку события в наши кнопки hideSessions и hideAgents, напримуер обрабатываемые функциями hideAll sessionsGroup и agentsGroup. Этот короткий кусочек кода позволит нам прятать всех агентов окон сессии при нажатии на нужную кнопку. В похожем ключе мы добавим больше ловушек событий в другие кнопки которые будут заставлять наши окна размещаться по экрану в очень простое обзорное окно:

```
g('tileAgents').on('click', this.tileAgents, this);
g('tileSessions').on('click', this.tileSessions, this);
```

на этот раз события обрабатываются tileAgents и tileSessions,что выглядит таким образом:

Исходный код примера:

Исходный код примера:

```
tileAgents : function(e) {
  this.sessionsGroup.hideAll();
  this.tile(this.agentsGroup);
},
tileSessions : function(e) {
  this.agentsGroup.hideAll();
  this.tile(this.sessionsGroup);
}
```

Повторим, мы вызываем функцию hideAll с целью удостовериться в том, что другой набор окон убран с экрана. Когда мы подготовили рабочее место, можно перейти к следующей функции:

Исходный код примера:

Исходный код примера:

```
tile : function(group) {
  var previousWin = null;
  group.each(function(win){
    if(previousWin) {
      if(win.getEl().getWidth() + previousWin.getEl().getRight() >
          Ext.getBody().getWidth()) {
        win.alignTo(document.body, 'tl-tl',
                    [0, previousWin.getEl().getHeight()]);
      } else {
        win.alignTo(previousWin.getEl(), 'tl-tr');
      }
    } else {
      win.alignTo(document.body, 'tl-tl');
    }
    previousWin = win;
  });
}
```

Здесь ключевой элемент, это функция WindowGroup.each, позволяющая перебирать окна назначенные в группу и выполнять с ними дальнейшие действия и расчеты. В этом случае, если это наш первая итерация, то мы выравниваем окна по телу документа. Для дальнейших, окна выравниваются по верху предыдущего окна в кольцо, установив вертикальный офсет, если окна будут сдвинуты в самую правую часть экрана.
hideAll и подобные, это полезные способы управления несколькими окнами. В этом примере мы показали как можно обработать две разные группы окона и при этом использовать минимум кода.

Заключение

Для меня Ext.Window является одним из фундаментальных аспектов наполненного приложения **Ext JS**. Большинство из других больших компонентов в оформлении, такие как таблица и дерево, могут отображать информацию, но если вам необходимо обработать и вывести пользователю большое количество информации без отвлечения его от работы, вам определенно понадобятся окна.
Самое главное, это то, что окна достаточно гибки и могут встраиваться непосредственно в приложение. Просто заполнить их своим или создать встроенный опыт, используя компоненты **Ext**. Подкласс панели окна дает возможность создавать сложные всплывающие окна используя наиболее подходящий Ext.layout и они выглядят точно так же как и другие части вашего приложения.
Хотя окна и диалоги с большой долей вероятности найдут свое место в вашем приложении, при их использовании следует соблюдать осторожность. Существует тенденция определять требование для отображения дополнительной информации и просто выдавать ее в лицо пользователю. Лучшим выходом было бы использование менее отвлекающего интерфейса. В заключение, Window и Mng располагаются чуть в стороне от других частей **Ext JS** и как инструменты могут быть легко встроены в любое приложение.