

[Обучение Ext JS] :: Часть 14. Сила Ext JS: Что еще можно сделать?

Мар 21 2010Другие форматирования

Пакет Ext.util дает много различных классов и методов для форматирования различных типов данных, включая различные ее строки, данные и числа. Хороший пример метода usMoney():

```
var annualSalary = 123456.345;
Ext.util.Format.usMoney(annualSalary);
```

Это вернет преформатированную строку валюты США с значением \$123,456.35. Может быть полезно при работе с приложениями занимающимися электронной коммерцией. Это один из многих методов внутри классов пакета Ext.util package.

Другой функцией является возможность извлекать HTML из строки. Скажем у нас есть простая форма для комментариев на странице, и мы хотим убедиться в том, что пользователь не будет вводить HTML-код, и наши данные не будут загрязнены перед попаданием на сервер.

Ext JS содержит простой метод для извлечения HTML из любой строки:

```
var firstName = '<b>jimi</b>';
var adj = Ext.util.Format.stripTags(firstName);
```

Мы можем также захотеть чтобы определенная информация, например имя, были написаны с заглавной буквы перед отправкой данных на сервер:

```
var adj2 = Ext.util.Format.capitalize(adj);
```

Также у нас может возникнуть желание применить значение по умолчанию к переменной если таковой нет:

```
var adj3 = Ext.util.Format.defaultValue(favoriteBlog,
    'Cutter\'s Crossing');
```

Это всего лишь несколько методов форматирования которые существуют в библиотеке Ext JS. Тщательное изучение API даст вам всю необходимую информацию.

Управление состоянием приложения

Ядром любого приложения (особенно управляемого событиями) является поддержание и управление "состоянием". Это может осуществить много разных вещей, от сохранения настроек пользователя при сортировке таблицы и до управления несколькими окнами или областями компоновки на экране, и даже возможность использовать кнопку возврата в браузере, изменяющую приложение без перезагрузки неправильной страницы. к счастью, Ext JS содержит несколько объектов Manager для обработки большинства из этих случаев.

Простое 'state'

Класс Ext.state.Manager автоматически проверяется и используется каждым компонентом имеющим дело с состояниями. Одной простой линией кода мы можем установить этот диспетчер и с небольшими (а то и без них) доработками состояние вида нашего приложения будет автоматически зарегистрировано.

Для того чтобы понять перспективу этого, представьте, что у нас есть довольно большое приложение предоставляющее различные документы HTML. Ext JS играет в приложении заметную роль, поскольку мы применили таблицу данных для отображения пользователей. Пока мы работаем над таблицей, может возникнуть необходимость изменить порядок сортировки со стоящего по умолчанию LastName на, скажем, UserName. Теперь, представим что у нас придется идти в редактор инвентаря. Если мы используем Manager, то когда мы вернемся к редактору пользователя, наша таблица будет автоматически отсортирована по Username, так как это было последнее сделанное нами изменение состояния:

```
Ext.state.Manager.setProvider(new Ext.state.CookieProvider());
```

Эта строка кода создает нашего диспетчера состояния, устанавливая поставщика для CookieProvider. Это значит, что состояние приложения сохранено в значении внутри cookie на клиентской машине, и которое в свою очередь будет прочитано для возобновления состояния при возвращении к компоненту.

ВАЖНО: CookieProvider также дает простой API для общего управления клиентскими cookies для наших сайтов, и может быть использован как отдельный класс ссылающийся и управляющий объектами cookie.

как мне получить это окно?

Большинство из приложений Ext JS привяаны к одной странце HTML, и могут быть просто набором объектов Ext JS Window на экране. Но что случится если пользователь занят одним окном, а мы хотим показать другое, которое уже открыто, без закрытия того с которым они работают в настоящий момент? По умолчанию окна Ext JS создаются и регистрируются внутри глобальной WindowGroup. Мы с легкостью можем управлять состоянием этих окон используя класс WindowMgr. Он позволяет нам брать любое окно из WindowGroup, и выносить его на передний план, отправлять на задний или даже прятать все окна группы.

Использование кнопки возврата в приложениях Ext JS

Одной общей проблемой среди Больших Интернет Приложений является обработка кнопки возврата в браузере. Предположим у нас есть большое, основанное на Ext JS приложение с закладками, таблицами и гармошками. Пользователь, переходя по приложению используя эти компоненты может нажать на кнопку возврата, чтобы вернуться к предыдущему действию, экрану или панели. Это ожидаемое поведение от большинства Интернет-приложений, поскольку они переходят с одной HTML страницы к другой.

Но большинство из Больших Интернет Приложений, как те что построены при помощи Ext JS, являются набором состояний внутри страницы HTML, а кнопка возврата отправляет пользователя к последней просмотренной им странице. К счастью, Ext JS 2.2 (последняя версия на момент написания этого) представила класс History, давая возможность управлять тем как используется история браузера, и как будет обрабатываться кнопка возврата внутри нашего приложения.

ВАЖНО: Существует несколько других классов Manager внутри Ext JS. StoreMgr дает простой доступ к различным объектам хранилища данных. ComponentMgr позволяет быстро получить особый компонент для работы. EventMgr позволяет управлять событиями внутри объектов Ext JS. Хороший обзор API Ext JS даст больше информации по этим диспетчерам и их использованию.

Работаем с DOM

Специальные адаптеры библиотеки позволят нам использовать Ext JS с другими библиотеками, например JQuery или Prototype. Но Ext JS дает собственные внутренние библиотеки для работы с DOM.

Ищем элементы DOM

Нашим первым заданием при управлении DOM будет найти то, что мы ищем. Класс DomQuery дает нам несколько методов для этого, включая возвращение всей группы узлов DOM которые подходят под критерий, или выбор единственного узла селектором. Мы можем начать поиск от особого узла на странице. Существует несколько типов селекторов, которые можно использовать для поиска элементов:

- простые селекторы элементов
- селекторы атрибутов
- псевдо-классы
- селекторы значения CSS

И более того, мы можем связать вместе несколько селекторов чтобы найти именно тот элемент, который мы ищем:

```
var myEl = Ext.DomQuery.selectNode
    ('a.iconLnk[@href*="cutterscrossing"]-first');
```

Это вернет первый якорный элемент с названием класса iconLnk, содержащий 'cutterscrossing' внутри атрибута href:

- a: якорный элемент
- .selectNode: Класс selectNode
- [@href *= "cutterscrossing"]: атрибут href содержащий 'cutterscrossing' внутри своего значения
- -first: Единственный первый элемент удовлетворяющий требованиям

Управление DOM

Класс DomHelper позволяет нам управлять DOM отрендеренной страницы на лету. Если мы хотим добавить новый элемент к набору остальных, удалить какой-либо элемент или даже заместить основное содержание элемента, то класс DomHelper содержит методы, которые помогут нам выполнить эти задачи:

```
Ext.DomHelper.insertAfter(ourGrid, newForm);
```

Эта линия разместит новый объект newForm непосредственно после ourGrid в DOM текущей страницы.

Можно с такой же легкостью использовать insertBefore() для размещения формы перед таблицей. Можно даже использовать insertHtml() чтобы вставить блок HTML внутрь DOM с особенным отношением к определенному элементу DOM. Тщательное изучение API даст понимание силы этого класса.

Работая со стилями

Класс DomHelper также дает простой метод для установки стиля для индивидуального элемента, через использование метода applyStyles(). но иногда нам надо изменить стиль всего документа. Для этого в Ext JS существует класс Ext.util.CSS, который позволяет нам создавать новые стили, которые будут автоматически применяться в начале документа, удаляя другие стили или меняя один стиль на другой. Мы даже можем действать по всему набору их правил:

```
Ext.util.CSS.swapStyleSheet('defaultDisplay','print.css');
```

Этот маленький скрипт заменит текущий стиль на стиль print.css.

Ext JS для рабочего стола: Adobe AIR

Одна из отличных вещей в том чтобы быть разработчиком веб-приложений, это возможность создавать по настоящему кросс-платформенные приложения. Традиционная модель для этого всегда было созданием браузерных приложений, которые работали в основном под парадигмой клиент/сервер с браузером в качестве клиента и сервером веб-приложения как сервер.

Ноу приложений рабочего стола есть большое количество преимуществ, включая доступ к собственной, локальной, файловой системе (то что не может быть осуществлено в веб-приложениях по соображениям безопасности). Самое трудное при создании десктопных приложений, это создание их для Windows, Unix/Linux, и Mac. Ни одна из этих систем не имеет общих библиотек отображения или доступа к файлам, таким образом создание кросс-платформенного приложения потребует отдельных версий кода для подгонки под каждую систему. Существуют языки, предоставляющие кросс-платформенные виртуальные машины, которые дают такой доступ к системным ресурсам, но им необходимо выучить язык, что может выходить за рамки инструментальной обшного бизнес-разработчика.

Adobe пришли к интересной идее. Почему бы не позволить людям разрабатывать кросс-платформенные приложения используя такие веб-технологии как HTML, JavaScript, AJAX и Flash?

После покупки Macromedia, Adobe получила прова на Flash player и платформу Flash. Flash player распространен среди 97% пользовательских десктопов в качестве плагина к большинству браузеров. Они также получили доступ к новой технологии под названием Flex, еоторая является основой и компилятором позволяющим разработчикам (а не дизайнерам) скриптовать приложения во Flash.

Но главное, они поняли как делегаты кросс-платформенный движок. Воспользовавшись новообретенным знанием они создали Adobe Integrated Runtime, более известный как AIR. AIR является десктопным движком для обработки создаваемого написанного в HTML, JavaScript, и/или Flash, что позволило получить локальный доступ к ресурсам клиентской системы, вне зависимости от типа операционной системы. Построенный на платформе WebKit, которая использовалась Apple в браузере Safari, AIR имеет доступ к локальной файловой системе и хранилищу, в то время как работая с веб-приложениями, таким образом создавая мостик между онлайн-содержимым и оффлайн-процессом. AIR дже дает доступ к локальной БД, созданной при помощи SQLite. SQLite это безсерверная библиотека для создания движка баз данных, который может быть непосредственно внедрен в приложение.


Adobe и Ext JS уже знакомы, поскольку Adobe работал с Ext чтобы можно было использовать библиотеку Ext JS как основу для компонентов работающих по AJAX-технологии созданными ColdFusion Markup Language, который работает на популярном сервере ColdFusion. И учитывая кросс-платформенную природу AIR, соответственно, что некоторые из первых образцов приложений HTML/JavaScript будут написаны при помощи другой кросс-платформенной библиотеки, Ext JS. В загрузке Ext JS это присутствует и будет отличным наглядным примером того, как создать приложения HTML и JavaScript для AIR.

Ext JS 2.1 была выпущена в тот же день что и Adobe AIR, и включает в себя весь пакет компонентов посвященных созданию приложений AIR основанных на Ext JS. Классы пакета Ext.air взаимодействуют с собственными классами AIR для взаимодействия с временем прогона и десктопом. Aptana, популярный редактор JavaScript и плагин Eclipse, добавляет поддержку для создания приложений AIR.

Для того чтобы сделать жизнь проще, простые класс Ext JS предоставляет логическое свойство isAir которое можно использовать для запуска приложений одновременно в сети и через приложение AIR, используя тот же самый код. Пакет Ext.air включает специальные пакеты для управления состоянием приложения, включая классы NativeWindowGroup и FileProvider (ответ AIR на WindowGroup и CookieProvider). А еще там есть классы для управления звуком и системным меню.

Дополнительная информация: Для дополнительной информации по разработке приложений Adobe AIR, посетите AIR and AJAX Developer Center (<http://www.adobe.com/devnet/air/ajax/>). Для подробностей посмотрите интеграцию AIR/Ext JS, изучите предоставляемых в загрузке Ext JS. Также существует соответствующая тема на форуме Ext JS.

- [« первая](#)
- [< предыдущая](#)
- [1](#)
- [2](#)
- [3](#)
- [следующая >](#)
- [последняя »](#)

-  [English](#)