AJAX и XML: Работа с таблицами в AJAX

Подготовили: Вавилов В.В. Скоробогатая М. А.

Введение

- Одной из сильных сторон технологии Ajax (Asynchronous JavaScript™ + XML) является динамическое отображение данных, полученных от сервера. Сейчас мы рассмотрим как описываются несколько методик для динамического представления данных при помощи таблиц, закладок и глайдеров.

Окна с закладками

- Закладки (tabs) предоставляют простейший способ того, как можно уместить множество данных на небольшой области экрана. Prototype – великолепная JavaScript-библиотека – до предела упрощает создание DHTML-окон с закладками и поддержкой Ajax.

Листинг 1

- Пример кода работы с Prototype.js

```
<html>
<head>
 <script src="prototype.js"></script>
</head>
<body>
    <a href="javascript:void loadTab( 'tab1.html' )">Tab 1</a> |
    <a href="javascript:void loadTab( 'tab2.html' )">Tab 2</a> |
    <a href="javascript:void loadTab( 'tab3.html' )">Tab 3</a>

    <div id="content" style="padding:5px;border:2px solid black;">
    </div>

    <script>
        function loadTab( tab ) {
          new Ajax.Updater( 'content', tab, { method: 'get' } );
        }
        loadTab( 'tab1.html' );
    </script>
</body>
</html>
```

Описание Листинга 1.

- В начале файла помещена ссылка на библиотеку Prototype.js, которая берет на себя всю работу с Ajax. Затем следует набор ссылок на различные страницы, причем нажатие на каждую ссылку приводит к вызову функции loadTab для обновления области, в которую загружается видимое содержимое. Эта область представляет собой элемент с идентификатором content. Функция loadTab вызывает метод Ajax.Updater для загрузки указанного документа HTML внутрь этого элемента

- Того же эффекта можно добиться при помощи скрытых элементов с предварительно загруженным содержимым, управляя их видимостью. Преимуществом подхода на основе Ajax является то, что содержимое закладок загружается только по мере необходимости. Таким образом снижается время на загрузку страницы, причем если пользователь решит не просматривать закладку, то ее содержимое не будет загружено вовсе.

Основы работы с таблицами в Ajax

- Вначале мы рассмотрим пример создания таблицы с отправкой XML-запроса на сервер при помощи Ajax. У этого метода есть два преимущества. Во-первых, он позволяет загружать данные по мере необходимости и обновлять их по месту, что улучшает восприятие пользовательского интерфейса. Во-вторых, для работы этого метода требуется источник XML-данных, который будет полезен не только в случае Ajax, но и при использовании любого клиентского кода, ожидающего данные в формате XML.

Код примера создания таблицы из XML-данных

```
<html>
<head>
<script src="prototype.js"></script>
</head>
<body>
<table id="books">
</table>
<script>
new Ajax.Request( 'books.xml', {
  method: 'get',
  onSuccess: function( transport ) {
    var bookTags = transport.responseXML.getElementsByTagName( 'book' );

    for( var b = 0; b < bookTags.length; b++ ) {
      var author = bookTags[b].getElementsByTagName('author')[0].firstChild.nodeValue;
      var title = bookTags[b].getElementsByTagName('title')[0].firstChild.nodeValue;
      var publisher =
          bookTags[b].getElementsByTagName('publisher')[0].firstChild.nodeValue;

      var elTR = $('books').insertRow( -1 );
      var elTD1 = elTR.insertCell( -1 );
      elTD1.innerHTML = author;
      var elTD2 = elTR.insertCell( -1 );
      elTD2.innerHTML = title;
      var elTD3 = elTR.insertCell( -1 );
      elTD3.innerHTML = publisher;
    }
  }
} );
</script>
</body>
</html>
```

- Обработчик события onSuccess при вызове Ajax.Request разбирает XML-данные, сначала выбирая элементы, описывающие книги. Затем выбираются значения вложенных тегов author, title и publisher. Наконец, обработчик добавляет данные в таблицу, вызывая методы insertRow и insertCell для каждой книги.

Пример формата XML, использующегося в этом примере

```xml
<books>
  <book>
    <author>Jack Herrington</author>
    <title>Code Generation In Action</title>
    <publisher>O'Reilly</publisher>
  </book>
  <book>
    <author>Jack Herrington</author>
    <title>Podcasting Hacks</title>
    <publisher>O'Reilly</publisher>
  </book>
  <book>
    <author>Jack Herrington</author>
    <title>PHP Hacks</title>
    <publisher>O'Reilly</publisher>
  </book>
</books>
```

Скрытые таблицы с разбиением на страницы

To be continued

```
<html>
<head>
<script src="prototype.js"></script>
</head>
<body>
<div>
    <?php
        require( 'data.php' );
        $data = getdata();
        $states = count( $data ) / 10;
        for( $s = 0; $s < $states; $s++ ) {
    ?>
    <?php echo( ( ( $s == 0 ) ? '' : '| ' ) ); ?>

    <a href="javascript:void updateTable(<?php echo( $s ) ?>);"> <?php echo( $s + 1 ) ?> </a> <?php } ?>
</div>

<?php $index = 0; foreach( $data as $state ) { ?>
<?php if ( $index % 10 == 0 ) { ?>
<?php if ( $index > 0 ) { ?> </table></div><?php } ?>

<div id="states<?php echo( $index / 10 ) ?>" style="display:none;">
    <table>
        <?php } ?> <tr>
        <?php foreach( $state as $item ) { ?>
        <td><?php echo($item); ?></td> <?php } ?> </tr>
        <?php $index += 1; } ?>
    </table>
</div>
```

Скрытые таблицы с разбиением на страницы

```
<script>
function updateTable( id )
{
  var elStateDivs = [];
<?php
for( $s = 0; $s < $states; $s++ ) {
?>
  elStateDivs.push( $( 'states<?php echo( $s ) ?>' ) );
<?php
}
?>
  for( var i = 0; i < elStateDivs.length; i++ ) {
    if ( i == id ) elStateDivs[i].show();
    else elStateDivs[i].hide();
  }
}

updateTable( 0 );
</script>
```

- В данном примере используется код на PHP для создания набора тегов , по одному на каждую страницу в таблице. Первый тег отображается по умолчанию, а остальные скрываются. Функция updateTable показывает и скрывает различные порции страницы в зависимости от выбранной страницы.
- Обратите внимание, что в примере по-прежнему используется библиотека Prototype.js, хотя поддержка Ajax не требуется. С ее помощью легче управлять элементам через предоставляемые ею методы $(),show и hide.

Глайдеры

- Для реализации эффекта сдвига страниц потребуется несколько дополнительных библиотек. Первой из них будет Scriptaculous – библиотека, реализованная на основе Prototype.js. Она предоставляет средства реализации эффектов, используемых глайдерами. Кроме нее будет также использоваться библиотека Glider.

Пример глайдера

```
<html><head>
<link rel="stylesheet" href="stylesheets/glider.css" type="text/css">
<script src="javascripts/prototype.js"></script>
<script src="javascripts/effects.js"></script>
<script src="javascripts/glider.js"></script>
</head><body>
<div id="glider"><div class="controls">
<a href="#tab1">Tab 1</a> |
<a href="#tab2">Tab 2</a> |
<a href="#tab3">Tab 3</a> |
<a href="#tab4">Tab 4</a></div>
<div class="scroller"><div class="content">
<div class="section" id="tab1">Tab 1</div>
<div class="section" id="tab2">Tab 2</div>
<div class="section" id="tab3">Tab 3</div>
<div class="section" id="tab4">Tab 4</div>
</div></div></div>
<script>
new Glider( 'glider', { duration:0.5 } );
</script>
</body></html>
```

- В начале страницы подключаются несколько скриптовых библиотек. Затем следует элемент глайдера, который содержит с идентификатором controls со ссылками на каждую закладку, а также другой с идентификатором scroller, в котором находится содержимое каждой закладки. Скрипт в нижней части страницы создает объект Glider с ID, равным идентификатору элемента глайдера

Заключение

- Нами было показано лишь несколько типов интерфейсных элементов, которые можно создавать при помощи Ajax, PHP и библиотеки Prototype.js. Надеюсь, некоторые из этих идей вы сможете применить в своих Web-приложениях. Они достаточно просты, причем Prototype.js действительно делает работу с Ajax тривиальной.

Спасибо за внимание