**ajax.url().load()**

1. **设置新的url并加载数据**
2. 使用[ajax.url()DT](http://datatables.club/reference/api/ajax.url().html)方法来设置新的请求地址， load()方法才会有效
3. functionajax.url().load( callback, resetPaging )

参数:

|  | 名称 | 类型 | 是否可选 |
| --- | --- | --- | --- |
| 1 | callback | [functionJS](http://datatables.net/reference/type/function) | Yes - default:null |
|  | 当服务器返回数据并重绘完毕时执行此回调方法，回调方法返回的是服务器返回的数据 | | |
| 2 | resetPaging | [booleanJS](http://datatables.net/reference/type/boolean) | Yes - default:true |
|  | 重置（默认或者设置为true）或者保持分页信息（设置为false） | | |

## 示例:

首先用data.json的数据初始化表格，然后立即用newData.json加载数据

var table = $('#example').DataTable( {

    ajax: "data.json"

} );

table.ajax.url( 'newData.json' ).load();

1. 下面我们来看DataTablesUtil用的这一个语句

DataTablesUtil.helper.loadUrl("XpFactoryUsergrid",url);

我们找到DataTablesUtil的helper属性的loadUrl属性：

/\*\*

\* 根据url重新加载表格数据

\*

\* @param id 表格id

\* @param url url

\*/

loadUrl: function (id, url) {

var $table = $("#" + id);

if ($table.attr("data-search")) {

$table.removeAttr("data-search");

}

**//把新的url赋给table的data-url属性**

$table.attr("data-url", url);

//dataTabelsDT哪里来的？

var $dt = $table.data("dataTablesDT");

if ($dt) {

$dt.ajax.url(url).load();

}

}

可以看到，通过获取table对象进行一系列解析，将我们自定义的url传入，然后table.ajax.url(url).load()异步设置table的data-url属性并通过URL加载数据

1. 为了知道ajax.url.load的细节，我们进入jquery.dataTables.js

我们知道，引入该js将自动启动工厂方法

(function( factory ) {

"use strict";

……

var DataTable = function ( options )

{

……

/\*\*

\* Load data from the newly set Ajax URL. Note that this method is only

\* available when `ajax.url()` is used to set a URL. Additionally, this method

\* has the same effect as calling `ajax.reload()` but is provided for

\* convenience when setting a new URL. Like `ajax.reload()` it will

\* automatically redraw the table once the remote data has been loaded.

\*

\* @returns {DataTables.Api} this

\*/

//这里注册这个方法

\_api\_register( '**ajax.url().load()**', function ( callback, resetPaging ){

// 跟reload类似,但注重于在url改变后表格数据的重新加载

return this.iterator( 'table', function ( ctx ) {

\_\_reload( ctx, resetPaging===false, callback );

} );

} );

可见，对url的处理还是没有展示，我们再看ajax.url

6.这里附上 ajax.url的源代码

/\*\*

\* Get the current Ajax URL. Note that this returns the URL from the first

\* table in the current context.

\*

\* @return {string} Current Ajax source URL

\*//\*\*

\* Set the Ajax URL. Note that this will set the URL for all tables in the

\* current context.

\*

\* @param {string} url URL to set.

\* @returns {DataTables.Api} this

\*/

\_api\_register( 'ajax.url()', function ( url ) {

var ctx = this.context;

//如果没有传入URL参数（表现为url===undefined）

if ( url === undefined ) {

// get

if ( ctx.length === 0 ) {

return undefined;

}

ctx = ctx[0];

//这里组装为一条URL

return ctx.ajax ?

$.isPlainObject( ctx.ajax ) ?

ctx.ajax.url :

ctx.ajax :

ctx.sAjaxSource;

}

// set

return this.iterator( 'table', function ( settings ) {

if ( $.isPlainObject( settings.ajax ) ) {

**settings.ajax.url = url;**

}

else {

**settings.ajax = url;**

}

// No need to consider sAjaxSource here since DataTables gives priority

// to `ajax` over `sAjaxSource`. So setting `ajax` here, renders any

// value of `sAjaxSource` redundant.

} );

} );

7.我们看到url被赋给了settings.ajax

于是我们找到了DataTables.settings的定义：

/\*\*

\* Private data store, containing all of the settings objects that are

\* created for the tables on a given page.

\*

\* Note that the `DataTable.settings` object is aliased to

\* `jQuery.fn.dataTableExt` through which it may be accessed and

\* manipulated, or `jQuery.fn.dataTable.settings`.

\* @member

\* @type array

\* @default []

\* @private

\*/

DataTable.settings = [];

意思是：私有的数据存储区域，其中包含了tables在页面创建时设定的那些设置属性对象，比如data-url等。

再深入一点就去jQuery.fn.dataTableExt和jQuery.fn.dataTable.settings找它（settings）的身影

我们要找ajax，于是我们就深入一点。