DWR – DIRECTED WEB REMOTING

Vamos ver neste artigo como personalizar uma tabela gerada pelo addRows().

Grids Personalizadas com addRow()

O DWR tem uma lib chamada *util.js,* que é nela que se localiza as principais funcionalidades do DWR. Vamos falar agora sobre o addRows() que tem como finalidade gerar grids(tabelas) dinâmicas. No artigo anterior vimos duas funções de callback do addRows(), são elas: ***rowCreator*** e ***cellCreator***.
É com elas que podemos alterar a tag html gerada pelo addRows para as tabelas.

Preparando o ambiente:

Aproveitando a estrutura dos artigos anteriores vamos fazer nesse artigo alguns exemplos de como personalizar nossa grid.

Utilizando o addRows().

Para os nossos exemplos vamos utilizar a seguinte tabela html(**Listagem 1**):

Listagem 01

```
<a href="javascript:carregaGridPersonalizada()" >Carregar</a>
<table border="0">
	<tr bgcolor="#CCCCCC">
		<td>Nome</td>
		<td>Endereço</td>
		<td>Telefone</td>
		<td>Cidade</td>
	</tr>
	<tbody id="gridPersonalizada"></tbody>
</table>
```

Criando uma função para verificar o browser.

Para utilizarmos alguns funções temos que primeiro ter uma função que verifique que tipo de browser estamos utilizando.

Mais você me pergunta: O DWR não é independente de Browser ?

Sim o Dwr é independente de browser, mais não esquecemos que ele é JavaScript, e vamos utilizar algumas funções do objeto DOM que é diferente em alguns browser, no decorrer do artigo você vai ver isso.

Listagem 02

```
/**
 * Função para validar o browser.
 * Retorna uma variável boolean.
 *
 * @return IE - boolean com true = IE / false = Mozilla
 */
function validaBrowser(){
	var IE = true;

	if(window.ActiveXObject){
		IE = true;
	}else{
		IE = false;
	}

	return IE;
}
```

A função acima(**Listagem 02**) testa se o tipo de objeto é do IE(Internet Explore) ou do Mozilla(Firefox, Opera, Netscape e etc).

Vamos agora ver alguns tipos de personalização.

Queremos adicionar um evento na linha da grid, por exemplo, quando a grid for gerada você poderá clicar na linha e ele executar uma ação qualquer.

Antes de vermos os exemplos tenha como base a seguinte função(**Listagem 03**):

Listagem 03

```
/**
* Função que acessa o método da classe java(FacadeAjax), e retorna *
como callback para a função montaXXX.
*/
function carregaGridPersonalizada(){
	FacadeAjax.selectAllPessoa(montaGridPersonalizadaX);
}
```

Essa função(**Listagem 03**) é responsável por acessar a classe java que vai retornar um List de Beans(List<Pessoa>) do banco, e terá como função de callback(retorno) os exemplos que iremos passar.

Exemplos:

Alterando a cor da linha

Vamos criar uma função que altere a cor da linha após essa mesma for clicada uma vez.

Listagem 04

```
function setColor(row){
	row.style.backgroundColor= '#ff0000';
}
```

O parâmetro (row) no código da **Listagem 04** é um objeto linha(<tr>), e o style.backgroundColor esta adicionando uma cor de fundo a linha.

Função principal addRows()

Listagem 05

```
function montaGridPersonalizada1(list){
	cleanAddRows("gridPersonalizada");

	var cellFuncs = [
		function(data) { return data.nome; },
		function(data) { return data.endereco; },
		function(data) { return data.telefone; },
		function(data) { return data.cidade; }
	];
```

```
	DWRUtil.addRows( "gridPersonalizada", list, cellFuncs, {
		rowCreator:function(options) {
			var row = document.createElement("tr");
			row.bgColor = "#CCCC99";

	if(validaBrowser()){
		row.attachEvent("onclick", function(e){setColor(row)});
	}else{
		row.addEventListener("click", function(e){setColor(row)}, false);
	}

	return row;
		},
		cellCreator:function(options) {
		var td = document.createElement("td");
			td.align = "center";

		return td;
		}
	});
}
```

Vamos entrar em detalhes apenas na personalização, caso você não entenda o funcionamento do addRows() aconselho a acessar o artigo DWR Parte 2.2

Explicando (Listagem 05):

Na função ***rowCreator*** vamos fazer um teste com o validaBrowser() **(Listagem 02)** para sabermos qual função vamos utilizar para adicionar na tag <TR> o ***onclick***.

Para o IE utilizamos o *attachEvent* que tem a seguinte assinatura:
attachEvent("evento", função);

Para os browser que seguem o padrão W3C como os baseados no Mozilla utilizamos o *addEventListener* que possui a seguinte assinatura:
addEventListener("evento", função, boolean);

Vimos aqui algumas diferenças.
No *attachEvent* colocamos o evento assim como no html, por exemplo:
onclick = onclick
onmousemove = onmousemove

Já no *addEventListener* temos que retirar o "ON" de cada evento, por exemplo:
ONclick = click
ONmousemove = mousemove

A função setColor(row) **(Listagem 04)** espera um objeto "row" que é criado em "`var row = document.createElement("tr");`" e o mesmo tem a propriedade "`style.backgroundColor`" alterada para "`#ff0000'`".

Quando clicamos na linha do meio da tabela temos o seguinte resultado exibido na imagem abaixo(**Imagem 01**):

Nome	Endereço	Telefone	Cidade
Christiano Milfont	Rua Rosas	32222222	Fortaleza
Handerson Frota	Rua Maximo	34444444	Fortaleza
Rafael Ponte	Rua Senhor dos Anéis	24242424	Fortaleza

Imagem 01

Apagando uma linha

Vamos fazer outro exemplo, agora queremos no evento "`ondblclick`" que ele apague a linha quando essa mesma foi clicada duas vezes. Vamos reutilizar o código da (**Listagem 05**), alterando apenas o seguinte trecho(**Listagem 06**), na realidade vamos adicionar um novo código(**ver Listagem 07**) em vermelho.

Listagem 06
```
function apagaLinha(row){
	row.style.display = 'none';
}
```

Listagem 07
```
if(validaBrowser()){
	row.attachEvent("onclick", function(e){setColor(row)});
	row.attachEvent("ondblclick", function(e){apagaLinha(row)});
}else{
	row.addEventListener("click", function(e){setColor(row)}, false);
	row.addEventListener("dblclick", function(e){apagaLinha(row)},
true);
}
```

A função da **Listagem 06**, seta na propriedade "`style.display`" o valor "`'none'`", que "esconde" a linha, dando a impressão que ela foi deletada.

Mais isso não é errado ? Você pergunta.

Não, na verdade quando você efetuar essa ação, você irá executar alguma regra, ela deleta do banco ou seja de onde for, e se tudo deu certo, a função esconde a linha, e caso o usuário aperte F5(atualizar) a grid será montada novamente e como esse valor foi deletado do banco ou de qualquer outro repositório, então ele não será exibido.

Grid criada, foi clicado uma vez na primeira e última linha(**Imagem 02**):

Nome	Endereço	Telefone	Cidade
Christiano Milfont	Rua Rosas	32222222	Fortaleza
Handerson Frota	Rua Maximo	34444444	Fortaleza
Rafael Ponte	Rua Senhor dos Anéis	24242424	Fortaleza

Imagem 02

Executamos o duplo click na linha do meio(**Imagem 03**):

Nome	Endereço	Telefone	Cidade
Christiano Milfont	Rua Rosas	32222222	Fortaleza
Rafael Ponte	Rua Senhor dos Anéis	24242424	Fortaleza

Imagem 03

Bem espero que o artigo tenha sido claro em mostrar o que é possível fazer com o DWR utilizando o addRows(), é claro que aqui mostramos exemplos simples, o poder das funcionalidades vai depender da sua criatividade e conhecimento em javascript, mais fica aqui a dica do que você poderá fazer e a liberdade que o DWR lhe dá.

E é claro que também podemos fazer com o <td>(colunas) utilizando o cellCreator que segue o mesmo padrão do rowCreator.

Como falei vai ficar a cargo da sua criatividade e necessidade, lembre-se que o addRows() vai gerar nada mais que HTML, e com o objeto DOM(javaScript) você tem total liberdade de mexer com o html gerado.

Boa sorte.

Até o próximo artigo.

Handerson Frota
(handersonbf@gmail.com)