DWR – DIRECTED WEB REMOTING

Dando continuidade ao nosso artigo vamos iniciar a segunda parte. Neste vamos abordar algumas funcionalidades um pouco mais avançadas do DWR.
Vamos também utilizar alguns exemplos com acesso ao banco de dados(MySql).

Parte 2 – Combos e Grids(addOptions(), addRows()).

Introdução

Vimos como instalar, configurar e rodar o DWR, agora vamos ver como é feito um preenchimento de uma combo utilizando o DWR.
Vamos utilizar um banco de dados(MySql), temos as seguintes tabelas de exemplo como foi citado no artigo anterior. Vemos agora a estrutura que iremos utilizar no MySql.

Nome do Banco: test

Tabela: pessoa	
Colunas:	**Tipo:**
Id	integer
Nome	varchar
Endereço	varchar
Cidade	varchar
Telefone	varchar

Tabela: usuário	
Colunas:	**Tipo:**
Id	int
Login	varchar
Senha	varchar

Sql para gerar o banco:
http://www.handersonfrota.com.br/artigos/dwr/combos/BancoArtigoDWR0061107_1509.sql

Agora o próximo passo será criar uma simples classe para conexão com o banco, chamaremos essa classe de ConectaBanco.java localizada no pacote `com.celulajava.palestradwr.dao`

Precisaremos também de algumas classes DAO's para executarem os sql's. Então criaremos os seguintes DAO's: DAOPessoa.java e DAOUsuario.java, que também estarão localizadas no pacote DAO.

Não vamos muito focar nos DAO's e nem no banco, apenas saiba que estaremos utilizando um banco de dados para recuperar valores.

Bem agora temos nosso ambiente pronto para acessarmos o banco de dados, vamos dar inicio então aos tão esperados exemplos.

Parte 2.1 – Combos (addOptions()).

Criando combos(DWRUtil.addOptions())

O DWR tem uma lib chamada *util.js,* que é nela que se localiza as principais funcionalidades do DWR. Vamos falar agora sobre o addOptions() que tem como finalidade gerar e popular uma comboBox(<select></select>), e/ou uma lista numerada ou de símbolos do html(
,).

Reutilizando a estrutura do nosso projeto anterior vamos criar um jsp chamado combos.jsp, e logo em seguida um link na nossa index.jsp que tenha acesso ao combos.jsp, criaremos também só por motivo didático um novo js, destinado apenas para as combos, vamos chama-lo de combos.js, veja abaixo:

Trecho de código na index.jsp.

```
<br>
<a href="combos.jsp">Exemplos de Combos</a>
```

(Figura 1)

Combo.jsp

combos.jsp

```
<html>
    <head>
        <script type="text/javascript" src='<%=request.getContextPath() %>/dwr/interface/FacadeAjax.js'></script>
        <script type="text/javascript" src='<%=request.getContextPath() %>/dwr/engine.js'></script>
        <script type="text/javascript" src='<%=request.getContextPath() %>/dwr/util.js'></script>
        <script type="text/javascript" src='js/combos.js'></script>
        <title>DWR: Ajax for Java - Handerson Frota - Portal Java Magazine</title>
    </head>

    <body>
        Página com exemplos do addOptions();<br><br>
    </body>
</html>
```

Observe que os importes das bibliotecas .js são os mesmo, exceto pelo novo importe, que é o do nosso combos.js.

Explicando o addOptions();

O addOptions é uma função da biblioteca util.js que tem a finalidade de gerar uma combo dinâmica utilizando o conceito de ájax e que rode em qualquer browser. Com ela você poderá gerar uma combo com dados de um banco, com o formato de arrays, map, list de beans e ainda gerar uma lista numerada.

Utilizando o addOptions:

Primeiro entenda a sintaxe de um select.

```
<select>
        <option value="valor">Texto</option>
</select>
```

1. **Array:** Recebe um simples Array com texto e valor.
    a. **DWRUtil.addOptions(selectid, array);**

2. **Array de Objetos (texto da <option> = valor da <option>):** Cria a combo a partir de um array de objetos, e definindo que o valor da <option> será o mesmo do texto da <option> do elemento <select>
    a. **DWRUtil.addOptions(selectid, arrayObjects, propriedade);**
    b. **Exemplo de Array de Objetos:** List de Beans, List de Maps, Collection etc.
    c. **Exemplo da option gerada:**
    DWRUtil.addOptions("combo", listBeans, "nome");
    <option value="Handerson">Handerson</option>

3. **Array de Objetos (texto da <option> != valor da <option>):** Cria a combo a partir de um array de objetos, definindo que o valor da <option> será diferente do texto da <option> do elemento <select>
    a. **DWRUtil.addOptions(selectid, arrayObjects, valorPropriedade, valorTexto);**
    b. **Exemplo de Array de Objetos:** List de Beans, List de Maps, Collection etc
    c. **Exemplo da option gerada:**
    DWRUtil.addOptions("combo", listBeans, "id", "nome");
    <option value="2">Handerson</option>

4. **Objeto:** Gera a combo a partir de um Objeto qualquer, se caso você utilize um Bean(Pessoa) com as seguintes propriedades(cidade, endereço, nome, telefone), ele irá carregar no campo value o valor da propriedade e no texto ele ira colocar o nome da propriedade.
    a. **DWRUtil.addOptions(selectid, arrayObjects, valorPropriedade);**
    b. **Exemplo de Objeto:** Beans
    c. **Exemplo da option gerada:**

DWRUtil.addOptions("combo", bean, "nome");
<option value="1">id</option>
<option value="Handerson">nome</option>
<option value="Fortaleza">cidade</option>
<option value="Rua Francisco">endereco</option>
<option value="3444444">telefone</option>

5. **Mapa de Objetos:** Gera uma combo a partir de um Map.
   a. **DWRUtil.addOptions(selectid, map);**
   b. **Exemplo de Objeto:** Map, Hashtable, HashMap.
   c. **Exemplo da option gerada:**
   DWRUtil.addOptions("combo", map);
   <option value="2">Handerson</option>

6. **Lista de <ol> ou <ul>:** Gera uma lista numerada ou com símbolo.
   a. **DWRUtil.addOptions(ul, list, valorPropriedade);**
   b. **Exemplo de Objeto:** List, Array, List de Beans.
   c. **Exemplo da option gerada:**
   DWRUtil.addOptions("ul", listBean, "nome");
   - Chirstiano Milfont
   - Handerson Frota

   Tag html: `<ul id="ul"></ul>`
   DWRUtil.addOptions("ol", listBean, "nome");
   1. Chirstiano Milfont
   2. Handerson Frota

   Tag html: `<ol id="ol"></ol>`

Agora para entendermos melhor, vamos visualizar como ficou na nossa camada do JAVA, que é o nosso FacadeAjax e DAO's, para facilitar a compreensão de como serão utilizado os acessos e chamadas dos nossos métodos, e saberemos também o tipo de retorno e como esta sendo criado para o envio ao nosso JS. Vamos utilizar os seguintes métodos para acessar o banco de dados.

FacadeAjax:

```
/**
 * Método que faz um simples select no banco e retorna um list de
 beans Pessoa.
 *
 * @return listResult - List de Object Bean Pessoa
 *
 * @throws Throwable
 * @throws Exception
 */
public List selectAllPessoa() throws Throwable, Exception{
      List<Pessoa> listResult = new ArrayList();

      listResult = DAOPessoa.selectAllPessoa();
```

```
        return listResult;
    }

    /**
     * Executa um select na tabela pessoa, mais so retorna um objeto
     Pessoa.
     *
     * @return pessoa - Object Bean Pessoa.
     * @throws Throwable
     * @throws Exception
     */
    public Pessoa selectPessoa() throws Throwable, Exception{
        List<Pessoa> listResult = new ArrayList();
        Pessoa pessoa = new Pessoa();

        listResult = DAOPessoa.selectAllPessoa();

        pessoa = listResult.get(0);

        return pessoa;
    }

    /**
     * Executa um select na tabela pessoa e retorna um Map com os
     *valores(id, nome)
     *
     * @return mapResult - Map com os dados do select.
     *
     * @throws Throwable
     * @throws Exception
     */
    public Map selectPessoaMap() throws Throwable, Exception{
        Map mapResult = new HashMap<Pessoa, Pessoa>();

        mapResult = DAOPessoa.selectPessoaMap();

        return mapResult;
    }
```

DAOPessoa:

```
    /**
     * Método que faz um simples select no banco e retorna um list de
     beans Pessoa
     *
     * @return dadosPessoaList List de Object Bean Pessoa
     * */
    public static List selectAllPessoa() throws Throwable, Exception{
        List<Pessoa> dadosPessoaList = new ArrayList();

      PreparedStatement stmt = null;
      ResultSet rs = null;

    stmt = conecta.getConnection().prepareStatement("select * from
    pessoa order by nome asc");
```

```
  rs = stmt.executeQuery();

  Pessoa pessoa = null;
  while(rs.next()){
      pessoa = new Pessoa();

      String id         = rs.getString("id");
      String nome       = rs.getString("nome");
            String endereco   = rs.getString("endereco");
            String cidade     = rs.getString("cidade");
            String telefone   = rs.getString("telefone");

            pessoa.setId(id);
            pessoa.setNome(nome);
            pessoa.setEndereco(endereco);
            pessoa.setCidade(cidade);
            pessoa.setTelefone(telefone);

      dadosPessoaList.add(pessoa);
      }

  stmt.close();
  rs.close();
  conecta.closeConnection();

      return dadosPessoaList;
}

/**
 * Executa um select na tabela pessoa e retorna um Map com os
 valores(id, nome)
 *
 * @return mapResult - Map com os dados do select.
 *
 * @throws Throwable
 * @throws Exception
 */
public static Map selectPessoaMap() throws Throwable, Exception{
      Map dadosPessoa = new HashMap();

  PreparedStatement stmt = null;
  ResultSet rs = null;

stmt = conecta.getConnection().prepareStatement("select * from
pessoa order by nome asc");

  rs = stmt.executeQuery();

  while(rs.next()){
      String id         = rs.getString("id");
      String nome       = rs.getString("nome");

      dadosPessoa.put(id, nome);
      }

  stmt.close();
```

```
            rs.close();
            conecta.closeConnection();

                return dadosPessoa;
        }
```

Como foi visto temos métodos que acessão o banco de dados, métodos simples apenas para exemplificar.

Bem agora que vimos os nossos métodos que iremos utilizar, vejamos os exemplos:

Criamos uma função JS para limpar as combos, sendo a mesma genérica, recebendo apenas o nome do select e/ou as tags de lista numerada que você deseja limpar:

```
function cleanAddOptions(local){
      DWRUtil.removeAllOptions(local);
}
```

Vou listar abaixo os exemplos equivalentes a explicação das assinaturas do addOptions(), e neles o respectivo código JSP e JS.

Exemplo 1 e 2(*Array, Array de Objetos (texto da <option> = valor da <option>)*):

JSP

```
<strong>1,2 - DWRUtil.addOptions("combo1_2", listBeans,
"nome");</strong><br>
Exemplo combo 1 e 2:
<a href="javascript:carregaCombo1_2()">Carregar</a> /
<a href="javascript:cleanAddOptions('combo1_2')" >Limpar</a>
<select id="combo1_2"></select>
```

JS

```
/**
 * Função que acessa o método da classe java(FacadeAjax), e retorna
 * como callback para a função montaXXX();
 */
function carregaCombo1_2(){
      FacadeAjax.selectAllPessoa(montaCombo1_2);
}

/**
 * Função de callback que limpa a combo e depois a remonta.
 * O processo para remover a combo é necessário se não os valores
 * seram somados com os já existentes.
 * @param {Object} listBeans
 */
function montaCombo1_2(listBeans){
      DWRUtil.removeAllOptions("combo1_2");
      DWRUtil.addOptions("combo1_2", listBeans, "nome");
}
```

Exemplo 3(*Array de Objetos (texto da <option> != valor da <option>):*):

JSP

```
<strong>3 - DWRUtil.addOptions("combo3", listBeans, "id",
"nome");</strong><br>
Exemplo combo 3:
<a href="javascript:carregaCombo3()" >Carregar</a> /
<a href="javascript:cleanAddOptions('combo3')" >Limpar</a>
<select id="combo3"></select>
```

JS

```
/**
 * Função que acessa o método da classe java(FacadeAjax), e retorna
 * como callback para a função montaXXX();
 */
function carregaCombo3(){
	FacadeAjax.selectAllPessoa(montaCombo3);
}

/**
 * Função de callback que limpa a combo e depois a remonta.
 * O processo para remover a combo é necessário se não os valores
 * seram somados com os já existentes.
 * @param {Object} listBeans
 */
function montaCombo3(listBeans){
	DWRUtil.removeAllOptions("combo3");
	DWRUtil.addOptions("combo3", listBeans, "id", "nome");
}
```

Exemplo 4(*Objeto*):

JSP

```
<strong>4 - DWRUtil.addOptions("combo4", bean, "nome");</strong><br>
Exemplo combo 4:
<a href="javascript:carregaCombo4()" >Carregar</a> /
<a href="javascript:cleanAddOptions('combo4')" >Limpar</a>
<select id="combo4"></select>
```

Até o próximo.

JS

```
/**
 * Função que acessa o método da classe java(FacadeAjax), e retorna
 * como callback para a função montaXXX();
 */
function carregaCombo4(){
	FacadeAjax.selectPessoa(montaCombo4);
}

/**
```

```
 * Função de callback que limpa a combo e depois a remonta.
 * O processo para remover a combo é necessário se não os valores
 * seram somados com os já existentes.
 * @param {Object} listBeans
 */
function montaCombo4(bean){
      DWRUtil.removeAllOptions("combo4");
      DWRUtil.addOptions("combo4", bean, "nome");
}
```

Exemplo 5(*Mapa de Objetos*):

JSP

```
<strong>5 - DWRUtil.addOptions("combo5", maps);</strong><br>
Exemplo combo 5:
<a href="javascript:carregaCombo5()" >Carregar</a> /
<a href="javascript:cleanAddOptions('combo5')" >Limpar</a>
<select id="combo5"></select>
```

JS

```
/**
 * Função que acessa o método da classe java(FacadeAjax), e retorna
 * como callback para a função montaXXX();
 */
function carregaCombo5(){
      FacadeAjax.selectPessoaMap(montaCombo5);
}

/**
 * Função de callback que limpa a combo e depois a remonta.
 * O processo para remover a combo é necessário se não os valores
 * seram somados com os já existentes.
 * @param {Object} listBeans
 */
function montaCombo5(maps){
      DWRUtil.removeAllOptions("combo5");
      DWRUtil.addOptions("combo5", maps);
}
```

Exemplo 6(*Lista de <ol> ou *):

JSP - <UL>

```
<strong>6 - DWRUtil.addOptions("ul", listBeans, "nome"); /
DWRUtil.addOptions("ol", listBeans, "nome");</strong><br>
Exemplo 6:<br>
Exemplo com a tag UL<br><a href="javascript:carregaUl()" >Carregar
UL</a> /
<a href="javascript:cleanAddOptions('ul')" >Limpar</a><br>
<ul id="ul"></ul>
```

JS

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

/**
 * Função de callback que limpa a combo e depois a remonta.
 * O processo para remover a combo é necessário se não os valores
 * seram somados com os já existentes.
 * @param {Object} listBeans
 */
function montaUl(listBeans){
      DWRUtil.removeAllOptions("ul");
      DWRUtil.addOptions("ul", listBeans, "nome");
}
```

JSP – <OL>

```
Exemplo com a tag OL<br><a href="javascript:carregaOl()" >Carregar
OL</a> /
<a href="javascript:cleanAddOptions('ol')" >Limpar</a><br>
<ol id="ol"></ol>
```

JS

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

/**
 * Função de callback que limpa a combo e depois a remonta.
 * O processo para remover a combo é necessário se não os valores
 * seram somados com os já existentes.
 * @param {Object} listBeans
 */
function montaOl(listBeans){
      DWRUtil.removeAllOptions("ol");
      DWRUtil.addOptions("ol", listBeans, "nome");
}
```

Estão listados acima as principais assinaturas do addOptions(). Podemos ver o quão fácil é gerar uma combo ou lista dinâmica com o DWR.

Para baixar esse projeto para executar os exemplos basta acessar o link abaixo:

http://www.handersonfrota.com.br/artigos/dwr/combos/PalestraDWR.rar

Veremos no próximo artigo(Parte 2.2) o addRows():

Handerson Frota
(handersonbf@gmail.com)