JSValueList

Apr 20, 2024 10:17

Supported Clients

SmartClient WebClient NGClient MobileClient

Constants Summary

Number	CUSTOM_VALUES	Constant to set the valueListType of a JSValueList.
Number	DATABASE_VALUES	Constant to set the valueListType of a JSValueList.
Number	EMPTY_VALUE_ALWAYS	Constant to set/get the addEmptyValue property of a JSValueList.
Number	EMPTY_VALUE_NEVER	Constant to set/get the addEmptyValue property of a JSValueList.

Property Summary

Number	addEmptyValue	Property that tells if an empty value must be shown next to the items in the value list.
String	customValues	A string with the elements in the valuelist.
String	dataSource	Compact representation of the names of the server and table that are used for loading the data from the database.
Number	displayValueType	
JSValueList	fallbackValueList	Gets or sets the fallback valuelist.
JSMethod	globalMethod	The global method of the valuelist is called to fill in or adjust the values of the valuelist.
Boolean	lazyLoading	A property special for NGClient and GlobalValuelist to only query the global valuelist when it is needed.
String	name	The name of the value list.
Number	realValueType	
String	relationName	The name of the relation that is used for loading data from the database.
String	separator	A String representing the separator that should be used when multiple display dataproviders are set, when the value list has the type set to database values.
String	serverName	The name of the database server that is used for loading the values when the value list has the type set to database/table values.
void	setFallbackValueList	
String	sortOptions	Sort options that are applied when the valuelist loads its data from the database.
String	tableName	The name of the database table that is used for loading the values when the value list has the type set to database/table values.
Boolean	useTableFilter	Flag that tells if the name of the valuelist should be applied as a filter on the 'valuelist_name' column when retrieving the data from the database.
Number	valueListType	The type of the valuelist.

Methods Summary

Array	getDisplayDataProviderIds()	Returns an array of the dataproviders that will be used to display the valuelist value.
Array	getReturnDataProviderIds()	Returns an array of the dataproviders that will be used to define the valuelist value that is saved.
UUID	getUUID()	Returns the UUID of the value list
void	setDisplayDataProviderIds()	Set the display dataproviders.
void	setDisplayDataProviderIds(dataprovider1)	Set the display dataproviders.
void	setDisplayDataProviderIds(dataprovider1, dataprovider2)	Set the display dataproviders.
void	setDisplayDataProviderIds(dataprovider1, dataprovider2, dataprovider3)	Set the display dataproviders.
void	setReturnDataProviderIds()	Set the return dataproviders.
void	setReturnDataProviderIds(dataprovider1)	Set the return dataproviders.
void	setReturnDataProviderIds(dataprovider1, dataprovider2)	Set the return dataproviders.
void	setReturnDataProviderIds(dataprovider1, dataprovider2, dataprovider3)	Set the return dataproviders.

Constants Details

CUSTOM_VALUES

```
Constant to set the valueListType of a JSValueList.
Sets the value list to use a custom list of values.
Also used in solutionModel.newValueList(...) to create new valuelists
```

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.DATABASE_VALUES);
vlist.valueListType = JSValueList.CUSTOM_VALUES; // Change the type to custom values.
vlist.customValues = "one\ntwo\nthree\nfour";
```

DATABASE_VALUES

Constant to set the valueListType of a JSValueList.
Sets the value list to use values loaded from a database.
Also used in solutionModel.newValueList(...) to create new valuelists

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.valueListType = JSValueList.DATABASE_VALUES; // Change the type to database values.
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text', 'parent_table_id');
vlist.separator = ' ## ';
vlist.sortOptions = 'parent_table_text desc';
```

EMPTY_VALUE_ALWAYS

Constant to set/get the addEmptyValue property of a JSValueList.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
vlist.addEmptyValue = JSValueList.EMPTY_VALUE_ALWAYS;
var cmb = form.newComboBox('my_table_text', 10, 10, 100, 20);
cmb.valuelist = vlist;
```

EMPTY_VALUE_NEVER

Constant to set/get the addEmptyValue property of a JSValueList.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
vlist.addEmptyValue = JSValueList.EMPTY_VALUE_NEVER;
var cmb = form.newComboBox('my_table_text', 10, 10, 100, 20);
cmb.valuelist = vlist;
```

Property Details

addEmptyValue

Property that tells if an empty value must be shown next to the items in the value list.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
vlist.addEmptyValue = JSValueList.EMPTY_VALUE_NEVER;
var cmb = form.newComboBox('my_table_text', 10, 10, 100, 20);
cmb.valuelist = vlist;
```

customValues

A string with the elements in the valuelist. The elements can be separated by linefeeds (custom1 custom2), optional with realvalues ((custom1|1 custom2|2)).

Returns

String

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var vl1 = solutionModel.newValueList("customtext",JSValueList.CUSTOM_VALUES);
vl1.customValues = "customvalue1\ncustomvalue2";
var vl2 = solutionModel.newValueList("customid",JSValueList.CUSTOM_VALUES);
vl2.customValues = "customvalue1|1\ncustomvalue2|2";
var form = solutionModel.newForm("customvaluelistform",controller.getDataSource(),null,true,300,300);
var combo1 = form.newComboBox("scopes.globals.text",10,10,120,20);
combo1.valuelist = vl1;
var combo2 = form.newComboBox("scopes.globals.id",10,60,120,20);
combo2.valuelist = vl2;
```

dataSource

Compact representation of the names of the server and table that are used for loading the data from the database.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.DATABASE_VALUES);
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text');
```

displayValueType

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

fallbackValueList

Gets or sets the fallback valuelist.

Returns

JSValueList

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var myValueList = solutionModel.getValueList('myValueListHere')
 //get fallback value list
var fallbackValueList = myValueList.fallbackValueList
```

globalMethod

The global method of the valuelist is called to fill in or adjust the values of the valuelist.
The method returns a dataset with one or two columns, first column is the display value, second column is real value(if present).
The valuelist will be filled in with the dataset data. If second column is not present real value and display value will be the same.
The method has to handle three different scenarios:
 1. 'displayValue' parameter is not null, this parameter should be used to filter the list of values(in a typeahead fashion)
 2. 'realValue' parameter is specified, that means value was not found in current list, so must be specified manually.
 In this case method should return only one row in the dataset, with the missing value, that will be added to the valuelist
 3. 'realValue' and 'displayValue' are both null , in this case the complete list of values should be returned.

Scenario 1 and 3 will completely replace any older results in the valuelist while scenario 2 will append results.

In find mode the record will be the FindRecord which is just like a normal JSRecord (DataRecord) it has the same properties (column/dataproviders) but doesnt have its methods (like isEditing())

The last argument is rawDisplayValue which contains the same text as displayValue but without converting it to lowercase.

Returns

JSMethod

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var listProvider = solutionModel.newGlobalMethod('globals', 'function getDataSetForValueList(displayValue,
realValue, record, valueListName, findMode, rawDisplayValue) {' +
                '        ' +
                'var args = null;' +
                'var query = datasources.db.example_data.employees.createSelect();' +
                '/** @type  {JSDataSet} */' +
                'var result = null;' +
                'query.result.add(query.columns.firstname.concat(' ').concat(query.columns.lastname)).add(query.
columns.employeeid);' +
                'if (displayValue == null && realValue == null) {' +
                '  // TODO think about caching this result. can be called often!' +
                '  // return the complete list' +
                '  result = databaseManager.getDataSetByQuery(query,100);' +
                '} else if (displayValue != null) {' +
                '  // TYPE_AHEAD filter call, return a filtered list' +
                '  args = [displayValue + "%", displayValue + "%"]' +
                '  query.result.root.where.add(query.or.add(query.columns.firstname.lower.like(args[0] + '%')).
add(query.columns.lastname.lower.like(args[1] + '%')));' +
                '  result = databaseManager.getDataSetByQuery(query,100);' +
                '} else if (realValue != null) {' +
                '  // TODO think about caching this result. can be called often!' +
                '  // real object not found in the current list, return 1 row with display,realvalue that will
be added to the current list' +
                '  // dont return a complete list in this mode because that will be added to the list that is
already there' +
                '  args = [realValue];' +
                '  query.result.root.where.add(query.columns.employeeid.eq(args[0]));' +
                '  result = databaseManager.getDataSetByQuery(query,1);' +
                '}' +
                'return result;' +
                '}');
var vlist = solutionModel.newValueList('vlist', JSValueList.CUSTOM_VALUES);
vlist.globalMethod = listProvider;
```

lazyLoading

A property special for NGClient and GlobalValuelist to only query the global valuelist when it is needed.
This flag has to be set both on valuelist and in component spec, on the valuelist property.

IMPORTANT: Usage of real & display values is not fully supported with lazy loading. Don't set lazy load if your method returns both real and display values
This is because very likely we do need directly the display value for the given real to dispaly its value..

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.DATABASE_VALUES);
vlist.dataSource = 'db:/example_data/valuelists';
vlist.setDisplayDataProviderIds('valuelist_data');
vlist.setReturnDataProviderIds('valuelist_data');
vlist.lazyLoading = true;
vlist.name = 'two';
```

name

The name of the value list.

It is relevant when the "useTableFilter" property is set.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.DATABASE_VALUES);
vlist.dataSource = 'db:/example_data/valuelists';
vlist.setDisplayDataProviderIds('valuelist_data');
vlist.setReturnDataProviderIds('valuelist_data');
vlist.useTableFilter = true;
vlist.name = 'two';
```

realValueType

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

relationName

The name of the relation that is used for loading data from the database.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var rel = solutionModel.newRelation('parent_to_child', 'db:/example_data/parent_table', 'db:/example_data
/child_table', JSRelation.INNER_JOIN);
rel.newRelationItem('parent_table_id', '=', 'child_table_parent_id');

var vlist = solutionModel.newValueList('options', JSValueList.DATABASE_VALUES);
vlist.dataSource = 'db:/example_data/parent_table';
vlist.relationName = 'parent_to_child';
vlist.setDisplayDataProviderIds('child_table_text');
vlist.setReturnDataProviderIds('child_table_text');
```

separator

A String representing the separator that should be used when multiple display dataproviders are set, when the value list has the type set to database values.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.valueListType = JSValueList.DATABASE_VALUES; // Change the type to database values.
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text', 'parent_table_id');
vlist.separator = ' ## ';
vlist.sortOptions = 'parent_table_text desc';
```

serverName

The name of the database server that is used for loading the values when the value list has the type set to database/table values.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.valueListType = JSValueList.DATABASE_VALUES; // Change the type to database values.
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text', 'parent_table_id');
vlist.separator = ' ## ';
vlist.sortOptions = 'parent_table_text desc';
```

setFallbackValueList

Supported Clients

SmartClient,WebClient,NGClient

Sample

sortOptions

Sort options that are applied when the valuelist loads its data
from the database.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.valueListType = JSValueList.DATABASE_VALUES; // Change the type to database values.
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text', 'parent_table_id');
vlist.separator = ' ## ';
vlist.sortOptions = 'parent_table_text desc';
```

tableName

The name of the database table that is used for loading the values when
the value list has the type set to database/table values.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.valueListType = JSValueList.DATABASE_VALUES; // Change the type to database values.
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text', 'parent_table_id');
vlist.separator = ' ## ';
vlist.sortOptions = 'parent_table_text desc';
```

useTableFilter

Flag that tells if the name of the valuelist should be applied as a filter on the
'valuelist_name' column when retrieving the data from the database.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.DATABASE_VALUES);
vlist.dataSource = 'db:/example_data/valuelists';
vlist.setDisplayDataProviderIds('valuelist_data');
vlist.setReturnDataProviderIds('valuelist_data');
vlist.useTableFilter = true;
vlist.name = 'two';
```

valueListType

The type of the valuelist. Can be either custom values or database values.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.valueListType = JSValueList.DATABASE_VALUES; // Change the type to database values.
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text', 'parent_table_id');
vlist.separator = ' ## ';
vlist.sortOptions = 'parent_table_text desc';
```

Methods Details

getDisplayDataProviderIds()

Returns an array of the dataproviders that will be used to display the valuelist value.

Returns

Array An array of Strings representing the names of the display dataproviders.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.DATABASE_VALUES);
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text', 'parent_table_id');
vlist.setReturnDataProviderIds('parent_table_text');
var dispDP = vlist.getDisplayDataProviderIds();
for (var i=0; i<dispDP.length; i++)
        application.output(dispDP[i]);
var retDP = vlist.getReturnDataProviderIds();
for (var i=0; i<retDP.length; i++)
        application.output(retDP[i]);
```

getReturnDataProviderIds()

Returns an array of the dataproviders that will be used to define the valuelist value that is saved.

Returns

Array An array of Strings representing the names of the return dataprovider.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.DATABASE_VALUES);
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text', 'parent_table_id');
vlist.setReturnDataProviderIds('parent_table_text');
var dispDP = vlist.getDisplayDataProviderIds();
for (var i=0; i<dispDP.length; i++)
        application.output(dispDP[i]);
var retDP = vlist.getReturnDataProviderIds();
for (var i=0; i<retDP.length; i++)
        application.output(retDP[i]);
```

getUUID()

Returns the UUID of the value list

Returns

UUID

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
application.output(vlist.getUUID().toString());
```

setDisplayDataProviderIds()

Set the display dataproviders. There can be at most 3 of them, combined with the return dataproviders. The values taken from these dataproviders, in order, separated by the separator, will be displayed by the valuelist.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.valueListType = JSValueList.DATABASE_VALUES; // Change the type to database values.
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text', 'parent_table_id');
vlist.separator = ' ## ';
vlist.sortOptions = 'parent_table_text desc';
```

setDisplayDataProviderIds(dataprovider1)

Set the display dataproviders. There can be at most 3 of them, combined with the return dataproviders. The values taken from these dataproviders, in order, separated by the separator, will be displayed by the valuelist.

Parameters

String dataprovider1 The first display dataprovider.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.valueListType = JSValueList.DATABASE_VALUES; // Change the type to database values.
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text', 'parent_table_id');
vlist.separator = ' ## ';
vlist.sortOptions = 'parent_table_text desc';
```

setDisplayDataProviderIds(dataprovider1, dataprovider2)

Set the display dataproviders. There can be at most 3 of them, combined with the return dataproviders. The values taken from these dataproviders, in order, separated by the separator, will be displayed by the valuelist.

Parameters

String dataprovider1 The first display dataprovider.
String dataprovider2 The second display dataprovider.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.valueListType = JSValueList.DATABASE_VALUES; // Change the type to database values.
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text', 'parent_table_id');
vlist.separator = ' ## ';
vlist.sortOptions = 'parent_table_text desc';
```

setDisplayDataProviderIds(dataprovider1, dataprovider2, dataprovider3)

Set the display dataproviders. There can be at most 3 of them, combined with the return dataproviders. The values taken from these dataproviders, in order, separated by the separator, will be displayed by the valuelist.

Parameters

String dataprovider1 The first display dataprovider.
String dataprovider2 The second display dataprovider.
String dataprovider3 The third display dataprovider.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.valueListType = JSValueList.DATABASE_VALUES; // Change the type to database values.
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text', 'parent_table_id');
vlist.separator = ' ## ';
vlist.sortOptions = 'parent_table_text desc';
```

setReturnDataProviderIds()

Set the return dataproviders. There can be at most 3 of them, combined with the display dataproviders. The values taken from these dataproviders, in order, separated by the separator, will be returned by the valuelist.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.valueListType = JSValueList.DATABASE_VALUES; // Change the type to database values.
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text', 'parent_table_id');
vlist.separator = ' ## ';
vlist.sortOptions = 'parent_table_text desc';
```

setReturnDataProviderIds(dataprovider1)

Set the return dataproviders. There can be at most 3 of them, combined with the display dataproviders. The values taken from these dataproviders, in order, separated by the separator, will be returned by the valuelist.

Parameters

String dataprovider1 The first return dataprovider.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.valueListType = JSValueList.DATABASE_VALUES; // Change the type to database values.
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text', 'parent_table_id');
vlist.separator = ' ## ';
vlist.sortOptions = 'parent_table_text desc';
```

setReturnDataProviderIds(dataprovider1, dataprovider2)

Set the return dataproviders. There can be at most 3 of them, combined with the display dataproviders. The values taken from these dataproviders, in order, separated by the separator, will be returned by the valuelist.

Parameters

String dataprovider1 The first return dataprovider.
String dataprovider2 The second return dataprovider.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.valueListType = JSValueList.DATABASE_VALUES; // Change the type to database values.
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text', 'parent_table_id');
vlist.separator = ' ## ';
vlist.sortOptions = 'parent_table_text desc';
```

setReturnDataProviderIds(dataprovider1, dataprovider2, dataprovider3)

Set the return dataproviders. There can be at most 3 of them, combined with the display dataproviders. The values taken from these dataproviders, in order, separated by the separator, will be returned by the valuelist.

Parameters

String dataprovider1 The first return dataprovider.
String dataprovider2 The second return dataprovider.
String dataprovider3 The third return dataprovider.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.valueListType = JSValueList.DATABASE_VALUES; // Change the type to database values.
vlist.dataSource = 'db:/example_data/parent_table';
vlist.setDisplayDataProviderIds('parent_table_text');
vlist.setReturnDataProviderIds('parent_table_text', 'parent_table_id');
vlist.separator = ' ## ';
vlist.sortOptions = 'parent_table_text desc';
```