JSDataSourceNode

Apr 05, 2024 12:49

Supported Clients

SmartClient WebClient NGClient

Methods Summary

JSCalculation	getCalculation(name)	Get an existing calculation for the datasource node.
Array	getCalculations()	Gets all the calculations for the datasource node.
String	getDataSource()	Get the data source for this node.
JSMethod	getMethod(name)	Get an existing foundset method for the datasource node.
Array	getMethods()	Gets all the foundset methods for the datasource node.
JSCalculation	newCalculation(code)	Creates a new calculation for the given code, the type will be the column where it could be build on (if name is a column name), else it will default to JSVariable.
JSCalculation	newCalculation(code, type)	Creates a new calculation for the given code and the type, if it builds on a column (name is a column name) then type will be ignored.
JSMethod	newMethod(code)	Creates a new foundset method with the specified code.
Boolean	removeCalculation(name)	Removes the calculation specified by name.
Boolean	removeMethod(name)	Removes the foundset method specified by name.

Methods Details

getCalculation(name)

Get an existing calculation for the datasource node.

Parameters

String name The name of the calculation

Returns

JSCalculation

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var calc = solutionModel.getDataSourceNode("db:/example_data/customers").newCalculation("function myCalculation()
{ return 123; }", JSVariable.INTEGER);
var calc2 = solutionModel.getDataSourceNode("db:/example_data/customers").newCalculation("function
myCalculation2() { return '20'; }");
var calc3 = solutionModel.getDataSourceNode("db:/example_data/employees").newCalculation("function
myCalculation3() { return 'Hello World!'; }",        JSVariable.TEXT);

var c = solutionModel.getDataSourceNode("db:/example_data/customers").getCalculation("myCalculation");
application.output("Name: " + c.getName() + ", Stored: " + c.isStored());

var allCalcs = solutionModel.getDataSourceNode("db:/example_data/customers").getCalculations();
for (var i = 0; i < allCalcs.length; i++) {
        application.output(allCalcs[i]);
}
```

getCalculations()

Gets all the calculations for the datasource node.

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var calc = solutionModel.getDataSourceNode("db:/example_data/customers").newCalculation("function myCalculation()
{ return 123; }", JSVariable.INTEGER);
var calc2 = solutionModel.getDataSourceNode("db:/example_data/customers").newCalculation("function
myCalculation2() { return '20'; }");
var calc3 = solutionModel.getDataSourceNode("db:/example_data/employees").newCalculation("function
myCalculation3() { return 'Hello World!'; }",        JSVariable.TEXT);

var c = solutionModel.getDataSourceNode("db:/example_data/customers").getCalculation("myCalculation");
application.output("Name: " + c.getName() + ", Stored: " + c.isStored());

var allCalcs = solutionModel.getDataSourceNode("db:/example_data/customers").getCalculations();
for (var i = 0; i < allCalcs.length; i++) {
        application.output(allCalcs[i]);
}
```

getDataSource()

Get the data source for this node.

Returns

String the dataSource

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var nodeDataSource = solutionModel.getDataSourceNode("db:/example_data/customers").getDataSource();
```

getMethod(name)

Get an existing foundset method for the datasource node.

Parameters

String name The name of the method

Returns

JSMethod

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var method = solutionModel.getDataSourceNode("db:/example_data/orders").newMethod("function doubleSize() {
return 2*getSize(); }");

application.output('Doubled orders for this customer: '+customers_to_orders.doubleSize())
```

getMethods()

Gets all the foundset methods for the datasource node.

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var method = solutionModel.getDataSourceNode("db:/example_data/orders").newMethod("function doubleSize() {
return 2*getSize(); }");

application.output('Doubled orders for this customer: '+customers_to_orders.doubleSize())
```

newCalculation(code)

Creates a new calculation for the given code, the type will be the column where it could be build on (if name is a column name),
else it will default to JSVariable.TEXT;

Parameters

String code The code of the calculation, this must be a full function declaration.

Returns

JSCalculation

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var calc = solutionModel.getDataSourceNode("db:/example_data/customers").newCalculation("function myCalculation()
{ return 123; }", JSVariable.INTEGER);
var calc2 = solutionModel.getDataSourceNode("db:/example_data/customers").newCalculation("function
myCalculation2() { return '20'; }");
var calc3 = solutionModel.getDataSourceNode("db:/example_data/employees").newCalculation("function
myCalculation3() { return 'Hello World!'; }",        JSVariable.TEXT);

var c = solutionModel.getDataSourceNode("db:/example_data/customers").getCalculation("myCalculation");
application.output("Name: " + c.getName() + ", Stored: " + c.isStored());

var allCalcs = solutionModel.getDataSourceNode("db:/example_data/customers").getCalculations();
for (var i = 0; i < allCalcs.length; i++) {
        application.output(allCalcs[i]);
}
```

newCalculation(code, type)

Creates a new calculation for the given code and the type, if it builds on a column (name is a column name) then type will be ignored.

Parameters

String code The code of the calculation, this must be a full function declaration.
Number type The type of the calculation, one of the JSVariable types.

Returns

JSCalculation

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var calc = solutionModel.getDataSourceNode("db:/example_data/customers").newCalculation("function myCalculation()
{ return 123; }", JSVariable.INTEGER);
var calc2 = solutionModel.getDataSourceNode("db:/example_data/customers").newCalculation("function
myCalculation2() { return '20'; }");
var calc3 = solutionModel.getDataSourceNode("db:/example_data/employees").newCalculation("function
myCalculation3() { return 'Hello World!'; }",        JSVariable.TEXT);

var c = solutionModel.getDataSourceNode("db:/example_data/customers").getCalculation("myCalculation");
application.output("Name: " + c.getName() + ", Stored: " + c.isStored());

var allCalcs = solutionModel.getDataSourceNode("db:/example_data/customers").getCalculations();
for (var i = 0; i < allCalcs.length; i++) {
        application.output(allCalcs[i]);
}
```

newMethod(code)

Creates a new foundset method with the specified code.

Parameters

String code the specified code for the foundset method

Returns

JSMethod a JSMethod object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var method = solutionModel.getDataSourceNode("db:/example_data/orders").newMethod("function doubleSize() {
return 2*getSize(); }");

application.output('Doubled orders for this customer: '+customers_to_orders.doubleSize())
```

removeCalculation(name)

Removes the calculation specified by name.

Parameters

String name the name of the calculation to be removed

Returns

Boolean true if the removal was successful, false otherwise

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var calc1 = solutionModel.getDataSourceNode("db:/example_data/customers").newCalculation("function
myCalculation1() { return 123; }", JSVariable.INTEGER);
var calc2 = solutionModel.getDataSourceNode("db:/example_data/customers").newCalculation("function
myCalculation2() { return '20'; }");

var c = solutionModel.getDataSourceNode("db:/example_data/customers").getCalculation("myCalculation1");
application.output("Name: " + c.getName() + ", Stored: " + c.isStored());

solutionModel.getDataSourceNode("db:/example_data/customers").removeCalculation("myCalculation1");
c = solutionModel.getDataSourceNode("db:/example_data/customers").getCalculation("myCalculation1");
if (c != null) {
        application.output("myCalculation could not be removed.");
}

var allCalcs = solutionModel.getDataSourceNode("db:/example_data/customers").getCalculations();
for (var i = 0; i < allCalcs.length; i++) {
        application.output(allCalcs[i]);
}
```

removeMethod(name)

Removes the foundset method specified by name.

Parameters

String name the name of the method to be removed

Returns

Boolean true if the removal was successful, false otherwise

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var method1 = solutionModel.getDataSourceNode("db:/example_data/customers").newMethod("function
myFoundsetMethod1() { return 123; }");
var method2 = solutionModel.getDataSourceNode("db:/example_data/customers").newCalculation("function
myFoundsetMethod2() { return '20'; }");

var m = solutionModel.getDataSourceNode("db:/example_data/customers").getMethod("myFoundsetMethod1");
application.output("Name: " + m.getName());

solutionModel.getDataSourceNode("db:/example_data/customers").removeMethod("myFoundsetMethod1");
m = solutionModel.getDataSourceNode("db:/example_data/customers").getCalculation("myFoundsetMethod1");
if (m != null) { application.output("myFoundsetMethod1 could not be removed."); }

var allMethods = solutionModel.getDataSourceNode("db:/example_data/customers").getMethod();
for (var i = 0; i < allMethods; i++)
{
        application.output(allMethods[i]);
}
```