

# JSRelation

For more information see [Relations](#)

## Constants Summary

**Number** [#INNER\\_JOIN](#)  
Constant for set/get the joinType of a JSRelation.

**Number** [#LEFT\\_OUTER\\_JOIN](#)  
Constant for set/get the joinType of a JSRelation.

## Property Summary

**Boolean** [#allowCreationRelatedRecords](#)  
Flag that tells if related records can be created through this relation.

**Boolean** [#allowParentDeleteWhenHavingRelatedRecords](#)  
Flag that tells if the parent record can be deleted while it has related records.

**Boolean** [#deleteRelatedRecords](#)  
Flag that tells if related records should be deleted or not when a parent record is deleted.

**String** [#foreignDataSource](#)  
Qualified name of the foreign data source.

**String** [#initialSort](#)  
A String which specified a set of sort options for the initial sorting of data retrieved through this relation.

**Number** [#joinType](#)  
The join type that is performed between the primary table and the foreign table.

**String** [#name](#)  
The name of the relation.

**String** [#primaryDataSource](#)  
Qualified name of the primary data source.

## Method Summary

**JSRelationItem[]** [#getRelationItems\(\)](#)  
Returns an array of JSRelationItem objects representing the relation criteria defined for this relation.

**UUID** [#getUUID\(\)](#)  
Returns the UUID of the relation object

**JSRelationItem** [#newRelationItem\(dataprovider, operator, foreinColumnName\)](#)  
Creates a new relation item for this relation.

**void** [#removeRelationItem\(primaryDataProviderID, operator, foreignColumnName\)](#)  
Removes the desired relation item from the specified relation.

## Constants Details

### INNER\_JOIN

Constant for set/get the joinType of a JSRelation. It is also used in solutionModel.newRelation(...).

#### Returns

Number

#### Sample

```
var relation = solutionModel.newRelation('parentToChild', 'example_data', 'parent_table', 'example_data',  
'child_table', JSRelation.INNER_JOIN);  
relation.joinType = JSRelation.LEFT_OUTER_JOIN;
```

### LEFT\_OUTER\_JOIN

Constant for set/get the joinType of a JSRelation. It is also used in solutionModel.newRelation(...).

#### Returns

Number

#### Sample

```
var relation = solutionModel.newRelation('parentToChild', 'example_data', 'parent_table', 'example_data',  
'child_table', JSRelation.INNER_JOIN);  
relation.joinType = JSRelation.LEFT_OUTER_JOIN;
```

## Property Details

### allowCreationRelatedRecords

Flag that tells if related records can be created through this relation.

The default value of this flag is "false".

#### Returns

Boolean

#### Sample

```
var relation = solutionModel.newRelation('parentToChild', 'example_data', 'parent_table', 'example_data',  
'child_table', JSRelation.INNER_JOIN);  
relation.allowCreationRelatedRecords = true;
```

### allowParentDeleteWhenHavingRelatedRecords

Flag that tells if the parent record can be deleted while it has related records.

The default value of this flag is "true".

#### Returns

Boolean

#### Sample

```
var relation = solutionModel.newRelation('parentToChild', 'example_data', 'parent_table', 'example_data',  
'child_table', JSRelation.INNER_JOIN);  
relation.allowParentDeleteWhenHavingRelatedRecords = false;
```

#### deleteRelatedRecords

Flag that tells if related records should be deleted or not when a parent record is deleted.

The default value of this flag is "false".

#### Returns

Boolean

#### Sample

```
var relation = solutionModel.newRelation('parentToChild', 'example_data', 'parent_table', 'example_data',
'child_table', JSRelation.INNER_JOIN);
relation.deleteRelatedRecords = true;
```

#### foreignDataSource

Qualified name of the foreign data source. Contains both the name of the foreign server and the name of the foreign table.

#### Returns

String

#### Sample

```
var relation = solutionModel.newRelation('parentToChild', 'example_data', 'parent_table', 'example_data',
'child_table', JSRelation.INNER_JOIN);
relation.primaryDataSource = 'db:/user_data/another_parent_table';
relation.foreignDataSource = 'db:/user_data/another_child_table';
```

#### initialSort

A String which specified a set of sort options for the initial sorting of data retrieved through this relation.

Has the form "column\_name asc, another\_column\_name desc, ...".

#### Returns

String

#### Sample

```
var relation = solutionModel.newRelation('parentToChild', 'example_data', 'parent_table', 'example_data',
'child_table', JSRelation.INNER_JOIN);
relation.initialSort = 'another_child_table_text asc';
```

#### joinType

The join type that is performed between the primary table and the foreign table.

Can be "inner join" or "left outer join".

#### Returns

Number

#### Sample

```
var relation = solutionModel.newRelation('parentToChild', 'example_data', 'parent_table', 'example_data',
'child_table', JSRelation.INNER_JOIN);
relation.joinType = JSRelation.LEFT_OUTER_JOIN;
```

#### name

The name of the relation.

#### Returns

String

#### Sample

```
var relation = solutionModel.newRelation('parentToChild', 'example_data', 'parent_table', 'example_data',
'child_table', JSRelation.INNER_JOIN);
relation.name = 'anotherName';
var firstTab = tabs.newTab('firstTab', 'Child Form', childForm, relation);
firstTab.relationName = relation.name;
```

#### primaryDataSource

Qualified name of the primary data source. Contains both the name of the primary server and the name of the primary table.

#### Returns

String

#### Sample

```
var relation = solutionModel.newRelation('parentToChild', 'example_data', 'parent_table', 'example_data',
'child_table', JSRelation.INNER_JOIN);
relation.primaryDataSource = 'db:/user_data/another_parent_table';
relation.foreignDataSource = 'db:/user_data/another_child_table';
```

## Method Details

#### getRelationItems

JSRelationItem[] **getRelationItems()**

Returns an array of JSRelationItem objects representing the relation criteria defined for this relation.

#### Returns

JSRelationItem[] – An array of JSRelationItem instances representing the relation criteria of this relation.

#### Sample

```
var criteria = relation.getRelationItems();
for (var i=0; i<criteria.length; i++)
{
    var item = criteria[i];
    application.output('relation item no. ' + i);
    application.output('primary column: ' + item.primaryDataProviderID);
    application.output('operator: ' + item.operator);
    application.output('foreign column: ' + item.foreignColumnName);
}
```

#### getUUID

UUID **getUUID()**

Returns the UUID of the relation object

#### Returns

UUID

#### Sample

```
var relation = solutionModel.newRelation('parentToChild', 'example_data', 'parent_table', 'example_data',
'child_table', JSRelation.INNER_JOIN);
application.output(relation.getUUID().toString())
```

#### newRelationItem

JSRelationItem **newRelationItem**(dataprovder, operator, foreinColumnName)

Creates a new relation item for this relation. The primary dataprovder, the foreign data provider and one relation operators (like '=' '!=' '>' '<') must be provided.

#### Parameters

{String} dataprovder – The name of the primary dataprovder.

{String} operator – The operator used to relate the primary and the foreign dataprovders.

{String} foreinColumnName – The name of the foreign dataprovder.

#### Returns

JSRelationItem – A JSRelationItem instance representing the newly added relation item.

#### Sample

```
var relation = solutionModel.newRelation('parentToChild', 'example_data', 'parent_table', 'example_data',
'child_table', JSRelation.INNER_JOIN);
relation.newRelationItem('another_parent_table_id', '=', 'another_child_table_parent_id');
```

#### removeRelationItem

void **removeRelationItem**(primaryDataProviderID, operator, foreignColumnName)

Removes the desired relation item from the specified relation.

**Parameters**

`{String}` primaryDataProviderID – the primary data provider (column) name

`{String}` operator – the operator

`{String}` foreignColumnName – the foreign column name

**Returns**

void

**Sample**

```
var relation = solutionModel.newRelation('myRelation', 'myServer', 'parentTable', 'myServer', 'childTable',
JSRelation.INNER_JOIN);
relation.newRelationItem('someColumn1', '=', 'someColumn2');
relation.newRelationItem('anotherColumn', '=', 'someOtherColumn');
relation.removeRelationItem('someColumn1', '=', 'someColumn2');
var criteria = relation.getRelationItems();
for (var i = 0; i < criteria.length; i++) {
    var item = criteria[i];
    application.output('primary column: ' + item.primaryDataProviderID);
    application.output('operator: ' + item.operator);
    application.output('foreign column: ' + item.foreignColumnName);
}
```