controller

Property Summary

Boolean	enabled Gets or sets the enabled state of a form; also known as "grayed-out".
Boolean	readOnly Gets or sets the read-only state of a form; also known as "editable" Note: The field(s) in a form set as read-only can be selected and the field data can be copied to clipboard.
Number	view Get/Set the current type of view of this form.

Method Summary

Boolean	deleteAllRecords() Deletes all records in foundset, resulting in empty foundset.
Boolean	deleteRecord() Delete current selected record, deletes mulitple selected records incase the foundset is using multiselect.
Boolean	duplicateRecord() Duplicate current record or record at index in the form foundset.
Boolean	duplicateRecord(location) Duplicate current record or record at index in the form foundset.
Boolean	duplicateRecord(location) Duplicate current record or record at index in the form foundset.
Boolean	find()
void	focusField(fieldName, skipReadonly) Sets focus to a field specified by its name.
void	focusFirstField() Sets focus to the first field of the form; based on tab order sequence.
Number	getDataProviderMaxLength(name) Returns the maximum length allowed in the specified dataprovider.
Object	getDataProviderValue(dataProvider) Gets a value based on the specified dataprovider name.
String	getDataSource() Get the used datasource.
Boolean	getDesignMode() Returns the state of this form designmode.
Object	getDesignTimeProperty() Get a design-time property of a form.
JSDataSet	getFormContext() Gets the forms context where it resides, returns a dataset of its structure to the main controller.
Number	getFormWidth() Gets the form width in pixels.
Number	getMaxRecordIndex() Returns the current cached record count of the current foundset.
String	getName() Get the name of this form.
Number	getPartHeight(partType) Gets the part height in pixels.
Number	getPartYOffset(partType) Returns the Y offset of a given part of the form.
Number	getSelectedIndex() Gets the current record index of the current foundset.
String[]	getTabSequence() Get an array with the names of the components that are part of the tab sequence.
JSWindow	getWindow() Returns the JSWindow that the form is shown in, or null if the form is not currently showing in a window.
Boolean	invertRecords() Inverts the current foundset against all rows of the current table; all records that are not in the foundset will become the current foundset.
Boolean	loadAllRecords() Loads all accessible records from the datasource into the form foundset.
Boolean	loadOmittedRecords() Loads the records that are currently omitted in the form foundset.
Boolean	loadRecords() Loads all accessible records from the datasource into the form foundset.
Boolean	loadRecords(foundset) Loads a (related) foundset into the form.

Boolean	loadRecords(pkdataset) Loads a primary key dataset, will remove related sort.
Boolean	loadRecords(UUIDpk) Loads a single record by primary key, will remove related sort.
Boolean	loadRecords(singlenNmber_pk) Loads a single record by primary key, will remove related sort.
Boolean	loadRecords(queryString) Loads records into form foundset based on a query (also known as 'Form by query').
Boolean	loadRecords(queryString, queryArgumentsArray) Loads records into form foundset based on a query (also known as 'Form by query').
Boolean	newRecord() Create a new record in the form foundset.
Boolean	newRecord(insertOnTop) Create a new record in the form foundset.
Boolean	newRecord(location) Create a new record in the form foundset.
Boolean	omitRecord() Omit current record in form foundset, to be shown with loadOmittedRecords.
void	print() Print this form with current foundset, without preview.
void	print(printCurrentRecordOnly) Print this form with current foundset, without preview.
void	print(printCurrentRecordOnly, showPrinterSelectDialog) Print this form with current foundset, without preview.
void	print(printCurrentRecordOnly, showPrinterSelectDialog, printerJob) Print this form with current foundset, without preview.
String	printXML() Print this form with current foundset records to xml format.
String	printXML(printCurrentRecordOnly) Print this form with current foundset records to xml format.
Boolean	recreateUI() Recreates the forms UI components, to reflect the latest solution model.
void	relookup() Performs a relookup for the current foundset record dataproviders.
Number	search()
Number	search(clearLastResults) Start the database search and use the results, returns the number of records, make sure you did "find" function first.
Number	search(clearLastResults, reduceSearch) Start the database search and use the results, returns the number of records, make sure you did "find" function first.
void	setDataProviderValue(dataprovider, value) Sets the value based on a specified dataprovider name.
void	setDesignMode(designMode) Sets this form in designmode with param true, false will return to normal browse/edit mode.
void	setDesignMode(ondrag) Sets this form in designmode with one or more callback methods.
void	setDesignMode(ondrag, ondrop) Sets this form in designmode with one or more callback methods.
void	setDesignMode(ondrag, ondrop, onselect) Sets this form in designmode with one or more callback methods.
void	setDesignMode(ondrag, ondrop, onselect, onresize) Sets this form in designmode with one or more callback methods.
void	setDesignMode(ondrag, ondrop, onselect, onresize, ondblclick) Sets this form in designmode with one or more callback methods.
void	setDesignMode(ondrag, ondrop, onselect, onresize, ondblclick, onrightclick) Sets this form in designmode with one or more callback methods.
void	setPageFormat(width, height, leftmargin, rightmargin, topmargin, bottommargin) Set the page format to use when printing.
void	setPageFormat(width, height, leftmargin, rightmargin, topmargin, bottommargin, orientation) Set the page format to use when printing.
void	setPageFormat(width, height, leftmargin, rightmargin, topmargin, bottommargin, orientation, units) Set the page format to use when printing.
void	setPreferredPrinter(printerName) Set the preferred printer name to use when printing.
void	setSelectedIndex(index) Sets the current record index of the current foundset.
void	setTabSequence(arrayOfElements) Set the tab order sequence programatically, by passing the elements references in a javascript array.
void	show() Shows the form (makes the form visible) This function does not affect the form foundset in any way.
void	show(window) Shows the form (makes the form visible) This function does not affect the form foundset in any way.
void	show(window) Shows the form (makes the form visible) This function does not affect the form foundset in any way.

void	showPrintPreview() Show this form in print preview.
void	showPrintPreview(printCurrentRecordOnly) Show this form in print preview.
void	showPrintPreview(printCurrentRecordOnly, printerJob) Show this form in print preview.
void	showPrintPreview(printCurrentRecordOnly, printerJob, zoomFactor) Show this form in print preview.
void	showRecords(foundset) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(foundset, window) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(foundset, window) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(pkdataset) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(pkdataset, window) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(pkdataset, window) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(query) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(query, window) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(query, window) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(UUIDpk) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(UUIDpk, window) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(UUIDpk, window) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(singleNumber_pk) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(singleNumber_pk, window) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(singleNumber_pk, window) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(query) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(query, window) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(query, argumentsArray) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(query, argumentsArray, window) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(query, argumentsArray, window) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	showRecords(query, window) Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.
void	sort(sortString) Sorts the form foundset based on the given sort string.
void	sort(sortString, defer) Sorts the form foundset based on the given sort string.
void	sortDialog() Show the sort dialog to the user a preselection sortString can be passed, to sort the form foundset.
void	sortDialog(sortString)

Property Details

enabled

Gets or sets the enabled state of a form; also known as "grayed-out".

Notes:
-A disabled element(s) cannot be selected by clicking the form.
-The disabled "grayed" color is dependent on the LAF set in the Servoy Smart Client Application Preferences.

Returns

Boolean

Sample

```
//gets the enabled state of the form
var state = forms.customer.controller.enabled;
//enables the form for input
forms.customer.controller.enabled = true;
```

readOnly

Gets or sets the read-only state of a form; also known as "editable"

Note: The field(s) in a form set as read-only can be selected and the field data can be copied to clipboard.

Returns

Boolean

Sample

```
//gets the read-only state of the form
var state = forms.customer.controller.readOnly;
//sets the read-only state of the form
forms.customer.controller.readOnly = true
```

view

Get/Set the current type of view of this form.

Returns

Number

Sample

```
//gets the type of view for this form
var view = forms.customer.controller.view;
//sets the form to Record view
forms.customer.controller.view = 0;//RECORD_VIEW
//sets the form to List view
forms.customer.controller.view = 1;//LIST_VIEW
```

Method Details

deleteAllRecords

Boolean **deleteAllRecords** ()

Deletes all records in foundset, resulting in empty foundset.

Returns

Boolean - false incase of related foundset having records and orphans records are not allowed by the relation

Sample

```
var success = forms.customer.controller.deleteAllRecords();
```

deleteRecord

Boolean **deleteRecord** ()

Delete current selected record, deletes mulitple selected records incase the foundset is using multiselect.

Returns

Boolean - false incase of related foundset having records and orphans records are not allowed by the relation

Sample

```
var success = forms.customer.controller.deleteRecord();
```

duplicateRecord

Boolean **duplicateRecord** ()

Duplicate current record or record at index in the form foundset.

Returns

Boolean - true if succesful

Sample

```
forms.customer.controller.duplicateRecord(); //duplicate the current record, adds on top
//forms.customer.controller.duplicateRecord(false); //duplicate the current record, adds at bottom
//forms.customer.controller.duplicateRecord(1,2); //duplicate the first record as second record
```

duplicateRecord

Boolean **duplicateRecord** (location)

Duplicate current record or record at index in the form foundset.

Parameters

{Boolean} location - true adds the new record as the topmost record

Returns

Boolean - true if successful

Sample

```
forms.customer.controller.duplicateRecord(); //duplicate the current record, adds on top
//forms.customer.controller.duplicateRecord(false); //duplicate the current record, adds at bottom
//forms.customer.controller.duplicateRecord(1,2); //duplicate the first record as second record
```

duplicateRecord

Boolean **duplicateRecord** (location)

Duplicate current record or record at index in the form foundset.

Parameters

{Number} location - adds at specified index

Returns

Boolean - true if successful

Sample

```
forms.customer.controller.duplicateRecord(); //duplicate the current record, adds on top
//forms.customer.controller.duplicateRecord(false); //duplicate the current record, adds at bottom
//forms.customer.controller.duplicateRecord(1,2); //duplicate the first record as second record
```

find

Boolean **find** ()

Returns

Boolean

Sample

```
if (forms.customer.controller.find()) //find will fail if autosave is disabled and there are unsaved records
{
        columnTextDataProvider = 'a search value'
        columnNumberDataProvider = '>10'
        columnDateDataProvider = '31-12-2010|dd-MM-yyyy'
        forms.customer.controller.search()
}
```

focusField

void **focusField** (fieldName, skipReadonly)

Sets focus to a field specified by its name.
If the second parameter is set to true, then readonly fields will be skipped
(the focus will be set to the first non-readonly field located after the field with the specified name; the tab sequence is respected when searching for the non-readonly field).

Parameters

{String} fieldName - the name of the field to be focussed
{Boolean} skipReadonly - indication to skip read only fields, if the named field happens to be read only

Returns

void

Sample

```
var tabseq = forms.customer.controller.getTabSequence();
if (tabseq.length > 1) {
        // If there is more than one field in the tab sequence,
        // focus the second one and skip over readonly fields.
        forms.customer.controller.focusField(tabseq[1], true);
}
else {
        // If there is at most one field in the tab sequence, then focus
        // whatever field is first, and don't bother to skip over readonly fields.
        forms.customer.controller.focusField(null, false);
}
```

focusFirstField

void **focusFirstField** ()

Sets focus to the first field of the form; based on tab order sequence.

Returns

void

Sample

```
forms.customer.controller.focusFirstField();
```

getDataProviderMaxLength

Number **getDataProviderMaxLength** (name)

Returns the maximum length allowed in the specified dataprovider.

Parameters

{String} name - the dataprovider name

Returns

Number - the length

Sample

```
forms.customer.controller.getDataProviderMaxLength('name');
```

getDataProviderValue

Object **getDataProviderValue** (dataProvider)

Gets a value based on the specified dataprovider name.

Parameters

{String} dataProvider - the dataprovider name to retieve the value for

Returns

Object - the dataprovider value (null if unknown dataprovider)

Sample

```
var val = forms.customer.controller.getDataProviderValue('contact_name');
```

getDataSource

String **getDataSource** ()

Get the used datasource.

Returns

String - the datasource

Sample

```
var dataSource = forms.customer.controller.getDataSource();
```

getDesignMode

Boolean **getDesignMode** ()

Returns the state of this form designmode.

Returns

Boolean - the design mode state (true/fase)

Sample

```
var success = forms.customer.controller.getDesignMode();
```

getDesignTimeProperty

Object **getDesignTimeProperty** ()

Get a design-time property of a form.

Returns

Object

Sample

```
var prop = forms.orders.controller.getDesignTimeProperty('myprop')
```

getFormContext

JSDataSet **getFormContext** ()

Gets the forms context where it resides, returns a dataset of its structure to the main controller.
Note1: can't be called in onload, because no context is yet available at this time.
Note2: tabindex is 1 (left) or 2 (right) for a SplitPane and 0 based for the other tabpanels; tabindex1based is the same as tabindex but is 1 based.

Returns

JSDataSet - the dataset with form context

Sample

```
//dataset columns: [containername(1),formname(2),tabpanel or beanname(3),tabname(4),tabindex(5),
tabindex1based(6)]
//dataset rows: mainform(1) -> parent(2)  -> current form(3) (when 3 forms deep)
/** @type {JSDataSet} */
var dataset = forms.customer.controller.getFormContext();
if (dataset.getMaxRowIndex() > 1)
{
        // form is in a tabpanel
        var parentFormName = dataset.getValue(1,2)
}
```

getFormWidth

Number **getFormWidth** ()

Gets the form width in pixels.

Returns

Number - the width in pixels

Sample

```
var width = forms.customer.controller.getFormWidth();
```

getMaxRecordIndex

Number **getMaxRecordIndex** ()

Returns the current cached record count of the current foundset.
To return the full foundset count, use: databaseManager.getFoundSetCount(...)
Tip: get the the table count of all rows in a table, use: databaseManager.getTableCount(...)

Returns

Number - the max record index

Sample

```
for ( var i = 1 ; i <= forms.customer.controller.getMaxRecordIndex() ; i++ )
{
        forms.customer.controller.setSelectedIndex(i);
        //do some action per record
}
```

getName

String **getName** ()

Get the name of this form.

Returns

String - the name

Sample

```
var formName = forms.customer.controller.getName();
```

getPartHeight

Number **getPartHeight** (partType)

Gets the part height in pixels.

Parameters

{Number} partType - The type of the part whose height will be returned.

Returns

Number - the part height in pixels

Sample

```
var height = forms.customer.controller.getPartHeight(JSPart.BODY);
```

getPartYOffset

Number **getPartYOffset** (partType)

Returns the Y offset of a given part of the form.

Parameters

{Number} partType - The type of the part whose Y offset will be returned.

Returns

Number - A number holding the Y offset of the specified form part.

Sample

```
var offset = forms.customer.controller.getPartYOffset(JSPart.BODY);
```

getSelectedIndex

Number **getSelectedIndex** ()

Gets the current record index of the current foundset.

Returns

Number - the index

Sample

```
//gets the current record index in the current foundset
var current = forms.customer.controller.getSelectedIndex();
//sets the next record in the foundset, will be reflected in UI
forms.customer.controller.setSelectedIndex(current+1);
```

getTabSequence

String[] **getTabSequence** ()

Get an array with the names of the components that are part of the tab sequence.
The order of the names respects the order of the tab sequence.
Components that are not named will not appear in the returned array, although they may be in the tab sequence.

Returns

String[] - array of names

Sample

```
var tabseq = forms.customer.controller.getTabSequence();
if (tabseq.length > 1) {
        // If there is more than one field in the tab sequence,
        // focus the second one and skip over readonly fields.
        forms.customer.controller.focusField(tabseq[1], true);
}
else {
        // If there is at most one field in the tab sequence, then focus
        // whatever field is first, and don't bother to skip over readonly fields.
        forms.customer.controller.focusField(null, false);
}
```

getWindow

JSWindow **getWindow** ()

Returns the JSWindow that the form is shown in, or null if the form is not currently showing in a window.

Returns

JSWindow - the JSWindow that the form is shown in, or null if the form is not currently showing in a window.

Sample

```
var currentWindow = controller.getWindow();
if (currentWindow != null) {
        currentWindow.title = 'We have a new title';
} else {
        currentWindow = application.createWindow("Window Name", JSWindow.WINDOW, null);
        currentWindow(650, 700, 450, 350);
        currentWindow = "Window Title";
        controller.show(currentWindow);
}
```

invertRecords

Boolean **invertRecords** ()

Inverts the current foundset against all rows of the current table; all records that are not in the foundset will become the current foundset.

Returns

Boolean - true if successful

Sample

```
forms.customer.controller.invertRecords();
```

loadAllRecords

Boolean **loadAllRecords** ()

Loads all accessible records from the datasource into the form foundset.
When the form contains a related foundset it will be replaced by a default foundset on same datasource.

Notes:
-the default foundset is always limited by filters, if databaseManager.addFoundSetFilterParam function is used.
-typical use is loading the normal foundset again after form usage in a related tabpanel

Returns

Boolean - true if successful

Sample

```
forms.customer.controller.loadAllRecords();
```

loadOmittedRecords

Boolean **loadOmittedRecords** ()

Loads the records that are currently omitted in the form foundset.

Returns

Boolean - true if successful

Sample

```
forms.customer.controller.loadOmittedRecords();
```

loadRecords

Boolean **loadRecords** ()

Loads all accessible records from the datasource into the form foundset. Typical usage is loading records after search in related tabpanel.
The difference to loadAllRecords() is that related foundset will load related records.

Returns

Boolean - true if successful

Sample

```
//to reload all last (related) records again, if for example after a search in related tabpanel
forms.customer.controller.loadRecords();
```

loadRecords

Boolean **loadRecords** (foundset)

Loads a (related) foundset into the form.
The form will no longer share the default foundset with forms of the same datasource, use loadAllRecords to restore the default foundset.

This will really update the foundset instance itself of the form, so now existing foundset is altered just the new foundset is shown.
This is different then doing foundset.loadRecords(foundset) because that just alters the current foundset and doesn't do anything with the foundset that is given.

Parameters

{JSFoundSet} foundset - to load

Returns

Boolean - true if successful

Sample

```
//to load a (related)foundset into the form.
//the form will no longer share the default foundset with forms of the same datasource, use loadAllRecords
to restore the default foundset
forms.customer.controller.loadRecords(order_to_orderdetails);
```

loadRecords

Boolean **loadRecords** (pkdataset)

Loads a primary key dataset, will remove related sort.

Parameters

{JSDataSet} pkdataset - to load

Returns

Boolean - true if successful

Sample

```
//to load a primary key dataset, will remove related sort
//var dataset = databaseManager.getDataSetByQuery(...);
// dataset must match the table primary key columns (alphabetically ordered)
forms.customer.controller.loadRecords(dataset);
```

loadRecords

Boolean **loadRecords** (UUIDpk)

Loads a single record by primary key, will remove related sort.

NOTE: This function will return true if the foundset was altered/changed. It is up to the developer to check for the presence of actual data using getSize().

Parameters

{UUID} UUIDpk - to load

Returns

Boolean - true if successful

Sample

```
forms.customer.controller.loadRecords(application.getUUID('6b5e2f5d-047e-45b3-80ee-3a32267b1f20'));
```

loadRecords

Boolean **loadRecords** (singlenNmber_pk)

Loads a single record by primary key, will remove related sort.

NOTE: This function will return true if the foundset was altered/changed. It is up to the developer to check for the presence of actual data using getSize().

Parameters

{Number} singlenNmber_pk - to load

Returns

Boolean - true if successful

Sample

```
forms.customer.controller.loadRecords(123);
```

loadRecords

Boolean **loadRecords** (queryString)

Loads records into form foundset based on a query (also known as 'Form by query'). The query must be a valid sql select.
If the foundset is related this function is not allowed.

see foundset.loadRecords(QBSelect).

Parameters

{String} queryString - to load

Returns

Boolean - true if successful

Sample

```
forms.customer.controller.loadRecords(sqlstring);
```

loadRecords

Boolean **loadRecords** (queryString, queryArgumentsArray)

Loads records into form foundset based on a query (also known as 'Form by query'). The query must be a valid sql select.
If the foundset is related this function is not allowed.

see foundset.loadRecords(QBSelect).

Parameters

{String} queryString - to load
{Object[]} queryArgumentsArray - the arguments to replace the questions marks in the queryString

Returns

Boolean - true if successful

Sample

```
forms.customer.controller.loadRecords(sqlstring,parameters);
```

newRecord

Boolean **newRecord** ()

Create a new record in the form foundset.

Returns

Boolean - true if succesful

Sample

```
// foreign key data is only filled in for equals (=) relation items
forms.customer.controller.newRecord();//default adds on top
//forms.customer.controller.newRecord(false); //adds at bottom
//forms.customer.controller.newRecord(2); //adds as second record
```

newRecord

Boolean **newRecord** (insertOnTop)

Create a new record in the form foundset.

Parameters

{Boolean} insertOnTop - true adds the new record as the topmost record

Returns

Boolean - true if successful

Sample

```
// foreign key data is only filled in for equals (=) relation items
forms.customer.controller.newRecord();//default adds on top
//forms.customer.controller.newRecord(false); //adds at bottom
//forms.customer.controller.newRecord(2); //adds as second record
```

newRecord

Boolean **newRecord** (location)

Create a new record in the form foundset.

Parameters

{Number} location - adds at specified index

Returns

Boolean - true if successful

Sample

```
// foreign key data is only filled in for equals (=) relation items
forms.customer.controller.newRecord();//default adds on top
//forms.customer.controller.newRecord(false); //adds at bottom
//forms.customer.controller.newRecord(2); //adds as second record
```

omitRecord

Boolean **omitRecord** ()

Omit current record in form foundset, to be shown with loadOmittedRecords.

Note: The omitted records are discarded when these functions are executed: loadAllRecords, loadRecords(dataset), loadRecords(sqlstring), invert

Returns

Boolean - true if successful

Sample

```
var success = forms.customer.controller.omitRecord();
```

print

void **print** ()

Print this form with current foundset, without preview.

Returns

void

Sample

```
//print this form (with foundset records)
forms.customer.controller.print();
//print only current record (no printerSelectDialog) to pdf plugin printer
//forms.customer.controller.print(true,false,plugins.pdf_output.getPDFPrinter('c:/temp/out.pdf'));
```

print

void **print** (printCurrentRecordOnly)

Print this form with current foundset, without preview.

Parameters

{Boolean} printCurrentRecordOnly - to print the current record only

Returns

void

Sample

```
//print this form (with foundset records)
forms.customer.controller.print();
//print only current record (no printerSelectDialog) to pdf plugin printer
//forms.customer.controller.print(true,false,plugins.pdf_output.getPDFPrinter('c:/temp/out.pdf'));
```

print

void **print** (printCurrentRecordOnly, showPrinterSelectDialog)

Print this form with current foundset, without preview.

Parameters

{Boolean} printCurrentRecordOnly - to print the current record only
{Boolean} showPrinterSelectDialog - to show the printer select dialog (default printer is normally used)

Returns

void

Sample

```
//print this form (with foundset records)
forms.customer.controller.print();
//print only current record (no printerSelectDialog) to pdf plugin printer
//forms.customer.controller.print(true,false,plugins.pdf_output.getPDFPrinter('c:/temp/out.pdf'));
```

print

void **print** (printCurrentRecordOnly, showPrinterSelectDialog, printerJob)

Print this form with current foundset, without preview.

Parameters

{Boolean} printCurrentRecordOnly - to print the current record only
{Boolean} showPrinterSelectDialog - to show the printer select dialog (default printer is normally used)
{PrinterJob} printerJob - print to plugin printer job, see pdf printer plugin for example

Returns

void

Sample

```
//print this form (with foundset records)
forms.customer.controller.print();
//print only current record (no printerSelectDialog) to pdf plugin printer
//forms.customer.controller.print(true,false,plugins.pdf_output.getPDFPrinter('c:/temp/out.pdf'));
```

printXML

String **printXML** ()

Print this form with current foundset records to xml format.

Returns

String - the XML

Sample

```
//TIP: see also plugins.file.writeXMLFile(...)
var xml = forms.customer.controller.printXML();
//print only current record
//var xml = forms.customer.controller.printXML(true);
```

printXML

String **printXML** (printCurrentRecordOnly)

Print this form with current foundset records to xml format.

Parameters

{Boolean} printCurrentRecordOnly - to print the current record only

Returns

String - the XML

Sample

```
//TIP: see also plugins.file.writeXMLFile(...)
var xml = forms.customer.controller.printXML();
//print only current record
//var xml = forms.customer.controller.printXML(true);
```

recreateUI

Boolean **recreateUI** ()

Recreates the forms UI components, to reflect the latest solution model.
Use this after altering the elements via solutionModel at the JSForm of this form.

Returns

Boolean - true if successful

Sample

```
// get the solution model JSForm
var form = solutionModel.getForm("myForm");
// get the JSField of the form
var field = form.getField("myField");
// alter the field
field.x = field.x + 10;
// recreate the runtime forms ui to reflect the changes.
forms.customer.controller.recreateUI();
```

relookup

void **relookup** ()

Performs a relookup for the current foundset record dataproviders.
Lookups are defined in the dataprovider (columns) auto-enter setting and are normally performed over a relation upon record creation.

Returns

void

Sample

```
forms.customer.controller.relookup();
```

search

Number **search** ()

Returns

Number

Sample

```
var recordCount = forms.customer.controller.search();
//var recordCount = forms.customer.controller.search(false,false); //to extend foundset
```

search

Number **search** (clearLastResults)

Start the database search and use the results, returns the number of records, make sure you did "find" function first.
Reduce results from previous searches.

Note: Omitted records are automatically excluded when performing a search - meaning that the foundset result by default will not include omitted records.

Parameters

{Boolean} clearLastResults - boolean, clear previous search, default true

Returns

Number - the recordCount

Sample

```
var recordCount = forms.customer.foundset.search();
//var recordCount = forms.customer.foundset.search(false,false); //to extend foundset
```

search

Number **search** (clearLastResults, reduceSearch)

Start the database search and use the results, returns the number of records, make sure you did "find" function first.

Note: Omitted records are automatically excluded when performing a search - meaning that the foundset result by default will not include omitted records.

Parameters

{Boolean} clearLastResults - boolean, clear previous search, default true
{Boolean} reduceSearch - boolean, reduce (true) or extend (false) previous search results, default true

Returns

Number - the recordCount

Sample

```
var recordCount = forms.customer.foundset.search();
//var recordCount = forms.customer.foundset.search(false,false); //to extend foundset
```

setDataProviderValue

void **setDataProviderValue** (dataprovider, value)

Sets the value based on a specified dataprovider name.

Parameters

{String} dataprovider - the dataprovider name to set the value for
{Object} value - the value to set in the dataprovider

Returns

void

Sample

```
forms.customer.controller.setDataProviderValue('contact_name','mycompany');
```

setDesignMode

void **setDesignMode** (designMode)

Sets this form in designmode with param true, false will return to normal browse/edit mode.

Parameters

{Boolean} designMode - sets form in design mode if true, false ends design mode.

Returns

void

Sample

```
var form = forms["selectedFormName"];
if (!form.controller.getDesignMode())
{
        // Set the current form in designmode with no callbacks
        form.controller.setDesignMode(true);
        // Set the current form in designmode with callbacks
        // where onDrag, onDrop, onSelect, onResize are names of form methods (not from "selectedFormName"
form)
        // form.controller.setDesignMode(onDrag, onDrop, onSelect, onResize);
}
//Set the current form out of designmode (to normal browse)
//form.controller.setDesignMode(false);
```

setDesignMode

void **setDesignMode** (ondrag)

Sets this form in designmode with one or more callback methods.

Parameters

{Function} ondrag - onDrag method reference

Returns

void

Sample

```
var form = forms["selectedFormName"];
if (!form.controller.getDesignMode())
{
        // Set the current form in designmode with no callbacks
        form.controller.setDesignMode(true);
        // Set the current form in designmode with callbacks
        // where onDrag, onDrop, onSelect, onResize are names of form methods (not from "selectedFormName"
form)
        // form.controller.setDesignMode(onDrag, onDrop, onSelect, onResize);
}
//Set the current form out of designmode (to normal browse)
//form.controller.setDesignMode(false);
```

setDesignMode

void **setDesignMode** (ondrag, ondrop)

Sets this form in designmode with one or more callback methods.

Parameters

{Function} ondrag - onDrag method reference
{Function} ondrop - onDrop method reference

Returns

void

Sample

```
var form = forms["selectedFormName"];
if (!form.controller.getDesignMode())
{
        // Set the current form in designmode with no callbacks
        form.controller.setDesignMode(true);
        // Set the current form in designmode with callbacks
        // where onDrag, onDrop, onSelect, onResize are names of form methods (not from "selectedFormName"
form)
        // form.controller.setDesignMode(onDrag, onDrop, onSelect, onResize);
}
//Set the current form out of designmode (to normal browse)
//form.controller.setDesignMode(false);
```

setDesignMode

void **setDesignMode** (ondrag, ondrop, onselect)

Sets this form in designmode with one or more callback methods.

Parameters

{Function} ondrag - onDrag method reference
{Function} ondrop - onDrop method reference
{Function} onselect - onSelect method reference

Returns

void

Sample

```
var form = forms["selectedFormName"];
if (!form.controller.getDesignMode())
{
        // Set the current form in designmode with no callbacks
        form.controller.setDesignMode(true);
        // Set the current form in designmode with callbacks
        // where onDrag, onDrop, onSelect, onResize are names of form methods (not from "selectedFormName"
form)
        // form.controller.setDesignMode(onDrag, onDrop, onSelect, onResize);
}
//Set the current form out of designmode (to normal browse)
//form.controller.setDesignMode(false);
```

setDesignMode

void **setDesignMode** (ondrag, ondrop, onselect, onresize)

Sets this form in designmode with one or more callback methods.

Parameters

{Function} ondrag - onDrag method reference
{Function} ondrop - onDrop method reference
{Function} onselect - onSelect method reference
{Function} onresize - onResize method reference

Returns

void

Sample

```
var form = forms["selectedFormName"];
if (!form.controller.getDesignMode())
{
        // Set the current form in designmode with no callbacks
        form.controller.setDesignMode(true);
        // Set the current form in designmode with callbacks
        // where onDrag, onDrop, onSelect, onResize are names of form methods (not from "selectedFormName"
form)
        // form.controller.setDesignMode(onDrag, onDrop, onSelect, onResize);
}
//Set the current form out of designmode (to normal browse)
//form.controller.setDesignMode(false);
```

setDesignMode

void **setDesignMode** (ondrag, ondrop, onselect, onresize, ondblclick)

Sets this form in designmode with one or more callback methods.

Parameters

{Function} ondrag - onDrag method reference
{Function} ondrop - onDrop method reference
{Function} onselect - onSelect method reference
{Function} onresize - onResize method reference
{Function} ondblclick - onDblClick method reference

Returns

void

Sample

```
var form = forms["selectedFormName"];
if (!form.controller.getDesignMode())
{
        // Set the current form in designmode with no callbacks
        form.controller.setDesignMode(true);
        // Set the current form in designmode with callbacks
        // where onDrag, onDrop, onSelect, onResize are names of form methods (not from "selectedFormName"
form)
        // form.controller.setDesignMode(onDrag, onDrop, onSelect, onResize);
}
//Set the current form out of designmode (to normal browse)
//form.controller.setDesignMode(false);
```

setDesignMode

void **setDesignMode** (ondrag, ondrop, onselect, onresize, ondblclick, onrightclick)

Sets this form in designmode with one or more callback methods.

Parameters

{Function} ondrag - onDrag method reference
{Function} ondrop - onDrop method reference
{Function} onselect - onSelect method reference
{Function} onresize - onResize method reference
{Function} ondblclick - onDblClick method reference
{Function} onrightclick - onRightClick method reference

Returns

void

Sample

```
var form = forms["selectedFormName"];
if (!form.controller.getDesignMode())
{
        // Set the current form in designmode with no callbacks
        form.controller.setDesignMode(true);
        // Set the current form in designmode with callbacks
        // where onDrag, onDrop, onSelect, onResize are names of form methods (not from "selectedFormName"
form)
        // form.controller.setDesignMode(onDrag, onDrop, onSelect, onResize);
}
//Set the current form out of designmode (to normal browse)
//form.controller.setDesignMode(false);
```

setPageFormat

void **setPageFormat** (width, height, leftmargin, rightmargin, topmargin, bottommargin)

Set the page format to use when printing.

Orientation values:
0 - Landscape mode
1 - Portrait mode

Units values:
0 - millimeters
1 - inches
2 - pixels

Note: The unit specified for width, height and all margins MUST be the same.

Parameters

{Number} width - the specified width of the page to be printed.
{Number} height - the specified height of the page to be printed.
{Number} leftmargin - the specified left margin of the page to be printed.
{Number} rightmargin - the specified right margin of the page to be printed.
{Number} topmargin - the specified top margin of the page to be printed.
{Number} bottommargin - the specified bottom margin of the page to be printed.

Returns

void

Sample

```
//Set page format to a custom size of 100x200 pixels with 10 pixel margins on all sides in portrait mode
forms.customer.controller.setPageFormat(100, 200, 10, 10, 10, 10);

//Set page format to a custom size of 100x200 pixels with 10 pixel margins on all sides in landscape mode
forms.customer.controller.setPageFormat(100, 200, 10, 10, 10, 10, SM_ORIENTATION.LANDSCAPE);

//Set page format to a custom size of 100x200 mm in landscape mode
forms.customer.controller.setPageFormat(100, 200, 0, 0, 0, 0, SM_ORIENTATION.LANDSCAPE, SM_UNITS.MM);

//Set page format to a custom size of 100x200 inch in portrait mode
forms.customer.controller.setPageFormat(100, 200, 0, 0, 0, 0, SM_ORIENTATION.PORTRAIT, SM_UNITS.INCH);
```

setPageFormat

void **setPageFormat** (width, height, leftmargin, rightmargin, topmargin, bottommargin, orientation)

Set the page format to use when printing.

Orientation values:
0 - Landscape mode
1 - Portrait mode

Units values:
0 - millimeters
1 - inches
2 - pixels

Note: The unit specified for width, height and all margins MUST be the same.

Parameters

{Number} width - the specified width of the page to be printed.
{Number} height - the specified height of the page to be printed.
{Number} leftmargin - the specified left margin of the page to be printed.
{Number} rightmargin - the specified right margin of the page to be printed.
{Number} topmargin - the specified top margin of the page to be printed.
{Number} bottommargin - the specified bottom margin of the page to be printed.
{Number} orientation - the specified orientation of the page to be printed; the default is Portrait mode

Returns

void

Sample

```
//Set page format to a custom size of 100x200 pixels with 10 pixel margins on all sides in portrait mode
forms.customer.controller.setPageFormat(100, 200, 10, 10, 10, 10);

//Set page format to a custom size of 100x200 pixels with 10 pixel margins on all sides in landscape mode
forms.customer.controller.setPageFormat(100, 200, 10, 10, 10, 10, SM_ORIENTATION.LANDSCAPE);

//Set page format to a custom size of 100x200 mm in landscape mode
forms.customer.controller.setPageFormat(100, 200, 0, 0, 0, 0, SM_ORIENTATION.LANDSCAPE, SM_UNITS.MM);

//Set page format to a custom size of 100x200 inch in portrait mode
forms.customer.controller.setPageFormat(100, 200, 0, 0, 0, 0, SM_ORIENTATION.PORTRAIT, SM_UNITS.INCH);
```

setPageFormat

void **setPageFormat** (width, height, leftmargin, rightmargin, topmargin, bottommargin, orientation, units)

Set the page format to use when printing.

Orientation values:
0 - Landscape mode
1 - Portrait mode

Units values:
0 - millimeters
1 - inches
2 - pixels

Note: The unit specified for width, height and all margins MUST be the same.

Parameters

{Number} width - the specified width of the page to be printed.
{Number} height - the specified height of the page to be printed.
{Number} leftmargin - the specified left margin of the page to be printed.
{Number} rightmargin - the specified right margin of the page to be printed.
{Number} topmargin - the specified top margin of the page to be printed.
{Number} bottommargin - the specified bottom margin of the page to be printed.
{Number} orientation - the specified orientation of the page to be printed; the default is Portrait mode
{Number} units - the specified units for the width and height of the page to be printed; the default is pixels

Returns

void

Sample

```
//Set page format to a custom size of 100x200 pixels with 10 pixel margins on all sides in portrait mode
forms.customer.controller.setPageFormat(100, 200, 10, 10, 10, 10);

//Set page format to a custom size of 100x200 pixels with 10 pixel margins on all sides in landscape mode
forms.customer.controller.setPageFormat(100, 200, 10, 10, 10, 10, SM_ORIENTATION.LANDSCAPE);

//Set page format to a custom size of 100x200 mm in landscape mode
forms.customer.controller.setPageFormat(100, 200, 0, 0, 0, 0, SM_ORIENTATION.LANDSCAPE, SM_UNITS.MM);

//Set page format to a custom size of 100x200 inch in portrait mode
forms.customer.controller.setPageFormat(100, 200, 0, 0, 0, 0, SM_ORIENTATION.PORTRAIT, SM_UNITS.INCH);
```

setPreferredPrinter

void **setPreferredPrinter** (printerName)

Set the preferred printer name to use when printing.

Parameters

{String} printerName - The name of the printer to be used when printing.

Returns

void

Sample

```
forms.customer.controller.setPreferredPrinter('HP Laser 2200');
```

setSelectedIndex

void **setSelectedIndex** (index)

Sets the current record index of the current foundset.

Parameters

{Number} index - the index to select

Returns

void

Sample

```
//gets the current record index in the current foundset
var current = forms.customer.controller.getSelectedIndex();
//sets the next record in the foundset, will be reflected in UI
forms.customer.controller.setSelectedIndex(current+1);
```

setTabSequence

void **setTabSequence** (arrayOfElements)

Set the tab order sequence programatically, by passing the elements references in a javascript array.

Parameters

{Object[]} arrayOfElements - array containing the element references

Returns

void

Sample

```
forms.customer.controller.setTabSequence([forms.customer.elements.fld_order_id, forms.customer.elements.
fld_order_amount]);
```

show

void **show** ()

Shows the form (makes the form visible)
This function does not affect the form foundset in any way.

Returns

void

Sample

```
// show the form in the current window/dialog
forms.customer.controller.show();
// show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.show(w);
// show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.show(w);
// or forms.customer.controller.show("mydialog");
//show the form in the main window
//forms.customer.controller.show(null);
```

show

void **show** (window)

Shows the form (makes the form visible)
This function does not affect the form foundset in any way.

Parameters

{JSWindow} window - the window in which this form should be shown, given as a window object

Returns

void

Sample

```
// show the form in the current window/dialog
forms.customer.controller.show();
// show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.show(w);
// show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.show(w);
// or forms.customer.controller.show("mydialog");
//show the form in the main window
//forms.customer.controller.show(null);
```

show

void **show** (window)

Shows the form (makes the form visible)
This function does not affect the form foundset in any way.

Parameters

{String} window - the window in which this form should be shown, specified by the name of an existing window

Returns

void

Sample

```
// show the form in the current window/dialog
forms.customer.controller.show();
// show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.show(w);
// show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.show(w);
// or forms.customer.controller.show("mydialog");
//show the form in the main window
//forms.customer.controller.show(null);
```

showPrintPreview

void **showPrintPreview** ()

Show this form in print preview.

Returns

void

Sample

```
//shows this form (with foundset records) in print preview
forms.customer.controller.showPrintPreview();
//to print preview current record only
//forms.customer.controller.showPrintPreview(true);
//to print preview current record only with 125% zoom factor;
//forms.customer.controller.showPrintPreview(true, null, 125);
```

showPrintPreview

void **showPrintPreview** (printCurrentRecordOnly)

Show this form in print preview.

Parameters

{Boolean} printCurrentRecordOnly - to print the current record only

Returns

void

Sample

```
//shows this form (with foundset records) in print preview
forms.customer.controller.showPrintPreview();
//to print preview current record only
//forms.customer.controller.showPrintPreview(true);
//to print preview current record only with 125% zoom factor;
//forms.customer.controller.showPrintPreview(true, null, 125);
```

showPrintPreview

void **showPrintPreview** (printCurrentRecordOnly, printerJob)

Show this form in print preview.

Parameters

{Boolean} printCurrentRecordOnly - to print the current record only
{PrinterJob} printerJob - print to plugin printer job, see pdf printer plugin for example (incase print is used from printpreview)

Returns

void

Sample

```
//shows this form (with foundset records) in print preview
forms.customer.controller.showPrintPreview();
//to print preview current record only
//forms.customer.controller.showPrintPreview(true);
//to print preview current record only with 125% zoom factor;
//forms.customer.controller.showPrintPreview(true, null, 125);
```

showPrintPreview

void **showPrintPreview** (printCurrentRecordOnly, printerJob, zoomFactor)

Show this form in print preview.

Parameters

{Boolean} printCurrentRecordOnly - to print the current record only
{PrinterJob} printerJob - print to plugin printer job, see pdf printer plugin for example (incase print is used from printpreview)
{Number} zoomFactor - a specified number value from 10-400

Returns

void

Sample

```
//shows this form (with foundset records) in print preview
forms.customer.controller.showPrintPreview();
//to print preview current record only
//forms.customer.controller.showPrintPreview(true);
//to print preview current record only with 125% zoom factor;
//forms.customer.controller.showPrintPreview(true, null, 125);
```

showRecords

void **showRecords** (foundset)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{JSFoundSet} foundset - the foundset to load before showing the form.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (foundset, window)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{JSFoundSet} foundset - the foundset to load before showing the form.
{JSWindow} window - the window in which this form should be shown, given as a window object.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (foundset, window)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{JSFoundSet} foundset - the foundset to load before showing the form.
{String} window - the window in which this form should be shown, specified by the name of an existing window.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (pkdataset)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{JSDataSet} pkdataset - the pkdataset to load before showing the form.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (pkdataset, window)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{JSDataSet} pkdataset - the pkdataset to load before showing the form.
{JSWindow} window - the window in which this form should be shown, given as a window object.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (pkdataset, window)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{JSDataSet} pkdataset - the pkdataset to load before showing the form.
{String} window - the window in which this form should be shown, specified by the name of an existing window.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (query)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{QBSelect} query - the query to load before showing the form.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (query, window)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{QBSelect} query - the query to load before showing the form.
{JSWindow} window - the window in which this form should be shown, given as a window object.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (query, window)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{QBSelect} query - the query to load before showing the form.
{String} window - the window in which this form should be shown, specified by the name of an existing window.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (UUIDpk)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{UUID} UUIDpk - the UUIDpk to load before showing the form.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (UUIDpk, window)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{UUID} UUIDpk - the UUIDpk to load before showing the form.
{JSWindow} window - the window in which this form should be shown, given as a window object.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (UUIDpk, window)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{UUID} UUIDpk - the UUIDpk to load before showing the form.
{String} window - the window in which this form should be shown, specified by the name of an existing window.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (singleNumber_pk)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{Number} singleNumber_pk - the singleNumber_pk to load before showing the form.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (singleNumber_pk, window)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{Number} singleNumber_pk - the singleNumber_pk to load before showing the form.
{JSWindow} window - the window in which this form should be shown, given as a window object

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (singleNumber_pk, window)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{Number} singleNumber_pk - the singleNumber_pk to load before showing the form.
{String} window - the window in which this form should be shown, specified by the name of an existing window.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (query)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{String} query - the query to load before showing the form.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (query, window)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{String} query - the query to load before showing the form.
{JSWindow} window - the window in which this form should be shown, given as a window object

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (query, argumentsArray)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{String} query - the query to load before showing the form.
{Object[]} argumentsArray - the array of arguments for the query

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (query, argumentsArray, window)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{String} query - the query to load before showing the form.
{Object[]} argumentsArray - the array of arguments for the query
{JSWindow} window - the window in which this form should be shown, given as a window object

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (query, argumentsArray, window)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{String} query - the query to load before showing the form.
{Object[]} argumentsArray - the array of arguments for the query
{String} window - the window in which this form should be shown, specified by the name of an existing window.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

showRecords

void **showRecords** (query, window)

Load data into the form and shows the form, is a shortcut for the functions 'loadRecords' and 'show'.

Parameters

{String} query - the query to load before showing the form.
{String} window - the window in which this form should be shown, specified by the name of an existing window.

Returns

void

Sample

```
forms.customer.controller.showRecords(foundset);
// load foundset & show the form in newly created named modal dialog
var w = application.createWindow("mydialog", JSWindow.MODAL_DIALOG);
forms.customer.controller.showRecords(foundset, w);
// load foundset & show the form in an existing window/dialog
var w = application.getWindow("mydialog"); // use null name for main app. window
forms.customer.controller.showRecords(foundset, w);
//forms.customer.controller.showRecords(foundset, "mydialog");
```

sort

void **sort** (sortString)

Sorts the form foundset based on the given sort string.
TIP: You can use the Copy button in the developer Select Sorting Fields dialog to get the needed syntax string for the desired sort fields/order.

Parameters

{String} sortString - the specified columns (and sort order)

Returns

void

Sample

```
forms.customer.controller.sort('columnA desc,columnB asc');
```

sort

void **sort** (sortString, defer)

Sorts the form foundset based on the given sort string.
TIP: You can use the Copy button in the developer Select Sorting Fields dialog to get the needed syntax string for the desired sort fields/order.

Parameters

{String} sortString - the specified columns (and sort order)
{Boolean} defer - the "sortString" will be just stored, without performing a query on the database (the actual sorting will be deferred until the next data loading action).

Returns

void

Sample

```
forms.customer.controller.sort('columnA desc,columnB asc');
```

sortDialog

void **sortDialog** ()

Show the sort dialog to the user a preselection sortString can be passed, to sort the form foundset.
TIP: You can use the Copy button in the developer Select Sorting Fields dialog to get the needed syntax string for the desired sort fields/order.

Returns

void

Sample

```
forms.customer.controller.sortDialog('columnA desc,columnB asc');
```

sortDialog

void **sortDialog** (sortString)

Parameters

{String} sortString - the specified columns (and sort order)

Returns

void

Sample

```
forms.customer.controller.sortDialog('columnA desc,columnB asc');
```