Grid

1. Grid

1.1. Aufrufreihenfolge

- init()
- get_columns()
- has_multi_selection()
- get_data()
- get_sortdata()
- get_selected()
- get_row_styles()
- get_actions()
- get_dblclick_actions()
- get_grouping()
- get_events()
- get_context_menu()

Die Methode get_tooltip() wird aufgerufen, wenn der Cursor sich auf einem entsprechenden Element befindet.

1.2. Util Actions und Platzhalter

> Diese Liste ist unvollständig.

1.2.1. Wichtige Util Actions

get_refresh_action

get_widget_set_action:

- Manipulierbare Felder ($property):
    - *filter* - Ändert den Filter des Grids, welcher über den Member $request erreichbar ist.
        - Bearbeiten$value ist ein assoziatives Array mit beliebigen Schlüssel-/Wertpaaren.
    - *query* - Ändert den Query des Grids, welcher über den Member $request erreichbar ist.
        - $value ist ein assoziatives Array mit beliebigen Schlüssel-/Wertpaaren.
    - *cell* - Ändert den Wert einer Zelle
        - $value ist ein Assoziatives Array welches *row_id* (ID des Datensatzes), *column_id* (ID der Spalte), *value* (Sichtbarer Wert) und *sort_value* (Sortierwert) enthält.

1.2.2. Platzhalter

Folgende Platzhalter für die Verwendung in den Util Actions sind verfügbar:

- *all_items* - Ermittelt alle Einträge. **Achtung: Es können nur maximal 1000 Werte mit einem Ajax-Request übermittelt werden, weitere Werte werden durch den Server abgeschnitten und *nicht* ausgewertet. Beispiel: Bei einem Grid mit 20 Spalten und 1000 Zeilen müssen 20000! Werte übertragen werden.**
- *all_items#column* - Ermittelt die Werte der angegebenen Spalte (column) für alle Einträge.
- *clicked_item* - Ermittelt den angeklickten Eintrag.
- *clicked_item#column* - Ermittelt den Wert der angegebenen Spalte der angeklickten Zeile.
- *event* - Ermittelt die Event-Daten einer Nutzerhandlung und übergibt diese. Beispielsweise wird bei einem Click-Event (vgl. get_events) die Position in Form von Zeilen und Spaltenangabe mit übermittelt. Die Daten werden als assoziatives Array in der folgenden Form übermittelt:
    - *column_id* - ID der Spalte
    - *item_id* - ID der Zeile
    - *value* - Wert der Zelle
- *filter#key* - Liefert einen Wert aus dem Filter zurück, welcher mit dem Schlüssel (key) identifiziert wird.
- *full_id* - Liefert die voll ID für den Zugriff auf das Widget zurück.
- *selected_items* - Ermittelt alle ausgewählten Einträge.
- *selected_items#column* - Ermittelt die Werte der angegebenen Spalte (column) für alle ausgewählten Einträge. Typischerweise wird die ID des Datensatzes benötigt, für welche die Spalte *item_id* angegeben werden muss.
- *selected_item* - Ermittelt den ersten ausgewählten Eintrag.
- *selected_item#column* - Ermittelt den Wert der angegebenen Spalte (column) für den ersten ausgewählten Eintrag. Typischerweise wird die ID des Datensatzes benötigt, für welche die Spalte *item_id* angegeben werden muss.
- *visible_columns* - Liefert alle für den Nutzer sichtbaren Spalten in der angezeigten Reihenfolge zurück.

1.3. Abstrakte Methoden

1.3.1. get_actions

Syntax:

array get_actions()

Beschreibung:

Erlaubt es Aktionen zu definieren, welche im Kontextmenü oder bei Doppelklick genutzt werden können.

Beispiel:

```
function get_actions()
{
  return array(
    'open' => addon_util_action::get_addon_window_action(
      'windows/edit',
      500,
      500,
      array('id' => '{self#item_id}', 'grid' => '{self#full_id}')
    ),
    'new' => addon_util_action::get_addon_window_action(
      'windows/add',
      500,
      500,
      array('id' => 0, 'grid' => '{self#full_id}')
    ),
    'send_selected' => addon_util_action::get_ajax_action(
      'my_ajax,
      array(
        'selected_items' => '{self#selected_items#item_id}'
      )
    ),
  );
}
```

1.3.2. get_columns

Syntax:

array get_columns()

Beschreibung:

Diese Methode muss ein Array aus Spalten zurückgeben. Eine Spalte wird jeweils durch ein assoziatives Array repräsentiert. Diesem können die folgenden Eigenschaften zugewiesen werden:

- *title* - Der Titel, der in der Kopfzeile der jeweiligen Spalte steht. Dieser wird, sofern eine Sprachvariable vorhanden ist, automatisch aufgelöst.
- *size* - Die initiale Spaltenbreite in Pixeln. Die Spaltenbreite kann nur initial gesetzt werden. Das heißt, dass nach dem ersten Aufruf des Grids durch einen Nutzer die Spaltenbreiten für diesen gespeichert werden und im nachhinein nicht durch den Entwickler verändert werden können.
- *align* - Die horizontale Ausrichtung des Inhalts und des Spaltenkopfes.
- *type* - Definiert die Formatierung des Inhalt der Spalte.
    - *string* - Einfacher Text
    - *date* - Datum
    - *number* - Zahl
    - *checkbox* - Checkbox
    - *link* - Visuelle Darstellung als Link
    - *module* - **(Ab v4.10)** Erstellt auf Basis von Objekt-IDs Verlinkungen zu einem Modul-Objekt. In den Daten für die Spalte dürfen nur Objekt-IDs hinterlegt werden. Die Angabe von Sortierdaten ist unnötig und wird ebenfalls automatisch befüllt. Um diesen Typen zu nutzen muss das Feld *module_id* angegeben sein.
- *field* - Der interne Bezeichner der Spalte, über den die Daten zugeordnet werden.
- *format* - Die Formatierung der Spalte, wenn vom Typ *date* oder *number*.
- *tooltip* - **(Ab v4.7)** Ein Tooltip, der angezeigt wird, wenn der Cursor sich über der Spalte im Header befindet.
- *hidden* - **(Ab v4.7)** Versteckt die Spalte. Der Nutzer kann die Spalte über das Header-Kontextmenü einblenden.
- *hideable* - **(Ab v4.7)** Bestimmt ob der Nutzer die Möglichkeit hat, die Spalte ein- bzw. auszublenden. Wenn *FALSE* und ein Wert für *hidden* gesetzt ist, bestimmt dieser Wert die Sichtbarkeit des Feldes. Die Einstellung des Nutzers wird dann ignoriert.
- *editable* - **(Ab v4.7)** Ermöglicht es, die Inhalte der Zellen der entsprechenden Spalte zu bearbeiten. **Hinweis:** Für Datumsfelder wird ein gewöhnlicher Editor genutzt und keine Datumsauswahl.
- *tooltip* - **(Ab v4.7)** Ermöglicht es, einen Tooltip anzugeben. Wenn Sie *tooltip* nicht angeben, versucht das Grid den Tooltip über die get_tooltip-Methode zu ermitteln
- *module_id* - **(Ab v4.10)** Definiert das Modul, für das Links erstellt werden sollen, wenn der Typ *module* verwendet wird. Zulässige Module sind Kontakte, Projekte, Verträge, Angebote, Tickets, Ressourcen, Artikel, Rechnungskonten, Ausgangsrechnungen, Mahnungen, Aufträge, Lieferungen, Gutschriften und Eingangsrechnungen. Die Module können mit den globalen Konstanten angegeben werden. Zu Kontakt-IDs kann zusätzlich eine Person mit einem Doppelpunkt getrennt angegeben werden. Diese Person muss mit dem Kontakt verknüpft sein.

Hinweis:

Die Reihenfolge wird durch den Entwickler initial bestimmt. Nutzer haben die Möglichkeit, Spalten zu verschieben. Die Reihenfolge kann anschließend nicht durch den Entwickler überschrieben werden. Neue Spalten werden hinten angefügt.

Formate:

Formate für den Typ *date*:

- *time_short* - Zeit kurz aus den Regionaleinstellungen
- *time_medium* - Zeit mittel aus den Regionaleinstellungen
- *time_long* - Zeit lang aus den Regionaleinstellungen
- *date_short* - Datum kurz aus den Regionaleinstellungen
- *date_medium* - Datum mittel aus den Regionaleinstellungen
- *date_long* - Datum lang aus den Regionaleinstellungen
- Weiter kann ein eigenes Format definiert werden, welches von date akzeptiert wird.

Wird kein Format angegeben, wird *time_long* verwendet.

Format für den Typ *number*:

Es kann ein eigenes Format definiert werden, welches von sprintf akzeptiert wird.

Gibt man kein Format an, wird das Zahlenformat aus dem CRM verwendet.

Beispiel:

```
function get_columns()
{
  return array(
    array('title' => 'Name', 'size' => 100, 'align' => 'left', 'type' => 'string', 'field' => 'name'),
    array('title' => 'Alter', 'size' => 40, 'align' => 'left', 'type' => 'number', 'field' => 'age',
'format' => '%.2f'),
    array('title' => 'Kontakt', 'size' => 150, 'align' => 'left', 'type' => 'module', 'field' => 'contact',
'module_id' => MODULE_CONTACTS),
  );
}
```

1.3.3. get_context_menu

Syntax:

array get_context_menu()

Beschreibung:

Muss ein Array aus Kontextmenüeinträgen zurückgeben. Ein einzelner Kontextmenüeintrag ist ein assoziatives Array, welches folgende Schüssel hat:

- *text* - Der für den Eintrag anzuzeigende Text. Wird wie jede Sprachvariable aufgelöst.
- *type* - **(Ab v4.7)** Typ des Eintrages. Derzeit stehen die Typen *default* (Ein gewöhnlicher Eintrag) und *separator* (Ein Abstandshalter) zur Verfügung. Default-Wert: *default*
- *actions* - Ein gewöhnliches Array, das die Schüssel der gewünschten Aktionen enthält, welche in get_actions beschrieben werden müssen.
- *id* - Die ID des Kontextmenüeintrages.
- *restriction* - optionale Einstellung für die Sichtbarkeit des Eintrages: (*single*: Nur bei Auswahl genau einer Zeile, *multiple*: Nur bei Auswahl mehrere Zeilen)

Beispiel:

```
function get_context_menu()
{
  return array(
    array('text' => 'Öffnen', 'actions' => array('open')),
    array('type' => 'separator' )), // Ab v4.7
    array('text' => 'Neu'   , 'actions' => array('new' )),
  );
}
```

1.3.4. get_data

Syntax:

array get_data()

Beschreibung:

Muss die Daten für das Grid bereitstellen. Die Daten werden in verschachtelten Arrays zweiten Grades angelegt. Das äußere Array enthält jeweils die gesamten Datensätze, während die inneren Arrays die Werte der einzelnen Datensätze enthalten. Die Keys der Werte müssen den Spalten entsprechend zugeordnet sein. Wichtig: Die Schlüssel der Arrays dürfen nicht den Wert 0 (Zahlenwert Null) enthalten.

Beispiel:

```
function get_data()
{
  return array(
    1 => array('name' => 'Bernhard', 'age' => '22', 'contact' => 35),
    2 => array('name' => 'Beatrice', 'age' => '42', 'contact' => 3632),
    3 => array('name' => 'Frauken' , 'age' => '32', 'contact' => 982690),
    4 => array('name' => 'Gustav'  , 'age' => '55', 'contact' => '353:56'),
  );
}
```

1.3.5. get_dblclick_action

Syntax:

array get_dblclick_action()

Beschreibung:

Muss ein Array bestehend aus Schlüsseln der Aktionen, die in get_actions beschrieben wurden, zurückgeben.

Beispiel:

```
function get_dblclick_action()
{
  return array('open');
}
```

1.3.6. get_events

Syntax:

array get_events()

Beschreibung:

Gibt ein assoziatives Array bestehend aus dem Event-Bezeichner als Schlüssel und dem Aktions-Bezeichner als Wert zurück. Die Aktionen werden in get_actions beschrieben. Zur Verfügung stehen die folgenden Events welche durch Klassen-Konstanten zu verwenden sind:

- *EVENT_CHANGE* - wird ausgelöst, wenn der Wert einer bearbeitbaren Zelle durch den Nutzer editiert wurde.
- *EVENT_CLICK* - wird ausgelöst, wenn der Nutzer eine Zeile im Grid anklickt.

Beispiel:

```
function get_events()
{
  return array(
    addon_grid::EVENT_CHANGE => 'on_change_action',
    addon_grid::EVENT_CLICK  => 'on_click_action'
  );
}
```

1.3.7. get_sortdata

Syntax:

array get_sortdata()

Beschreibung:

Muss die Sortierdaten für das Grid bereitstellen. Die Daten werden in verschachtelten Arrays zweiten Grades angelegt. Das äußere Array enthält jeweils die gesamten Datensätze, während die inneren Arrays die Werte der einzelnen Datensätze enthalten. Die Keys der Werte müssen den Spalten entsprechend zugeordnet sein.

Beispiel:

```
function get_sortdata()
{
  return array(
    1 => array('name' => 'M-Bernhard', 'age' => 22),
    2 => array('name' => 'F-Beatrice', 'age' => 42),
    3 => array('name' => 'F-Frauken' , 'age' => 32),
    4 => array('name' => 'M-Gustav'  , 'age' => 55),
  );
}
```

1.3.8. get_row_styles

Syntax:

array get_row_styles()

Beschreibung:

Muss ein Array bestehend aus Style-Klasse zurückgeben. Diese Klassen sind als Konstanten in der Klasse addon_utils definiert und können nach belieben kombiniert werden. Die folgenden Klassen stehen zur Verfügung:

- STYLE_FONTCOLOR_NAVY
- STYLE_FONTCOLOR_BLUE
- STYLE_FONTCOLOR_AQUA

- STYLE_FONTCOLOR_TEAL
- STYLE_FONTCOLOR_OLIVE
- STYLE_FONTCOLOR_GREEN
- STYLE_FONTCOLOR_LIME
- STYLE_FONTCOLOR_YELLOW
- STYLE_FONTCOLOR_ORANGE
- STYLE_FONTCOLOR_RED
- STYLE_FONTCOLOR_MAROON
- STYLE_FONTCOLOR_FUCHSI
- STYLE_FONTCOLOR_PURPLE
- STYLE_FONTCOLOR_BLACK
- STYLE_FONTCOLOR_GRAY
- STYLE_FONTCOLOR_SILVER
- STYLE_FONTCOLOR_WHITE
- STYLE_BOLD - Dicke Schrift.
- STYLE_ITALIC - Kursive Schrift.

Beispiel:

```
function get_row_styles()
{
  return array(
    addon_utils::STYLE_FONTCOLOR_BLUE . ' ' . addon_utils::STYLE_BOLD,
    addon_utils::STYLE_FONTCOLOR_RED,
    addon_utils::STYLE_FONTCOLOR_RED,
    addon_utils::STYLE_FONTCOLOR_BLUE,
  );
}
```

1.4. Optionale Methoden

1.4.1. get_fixed_left

Syntax:

int get_fixed_left()

Beschreibung:

Erlaubt **ab V4.7** Spalten von links aus gesehen im Grid zu fixieren. Diese bleiben beim horizontalen Scrollen stehen.

Beispiel:

```
function get_fixed_left() : int
{
  return 2;
}
```

1.4.2. get_fixed_right

Syntax:

int get_fixed_right()

Beschreibung:

Erlaubt **ab V4.7** Spalten von rechts aus gesehen im Grid zu fixieren. Diese bleiben beim horizontalen Scrollen stehen.

Beispiel:

```
function get_fixed_right() : int
{
  return 1;
}
```

1.4.3. get_grouping

Syntax:

array get_grouping()

Beschreibung:

Ab V4.7

Muss ein Array mit Gruppierungsinformationen zurückgeben.

Beispiel:

```
function get_grouping()
{
  return array('column' => 'lender', 'direction' => 'asc');
}
```

1.4.4. get_selected

Syntax:

?array get_selected()

Beschreibung:

(Ab v4.10) Bestimmt auf Basis der übergebenen IDs, welche Einträge ausgewählt sein sollen. Bei der Rückgabe eines leeren Arrays wird kein Eintrag ausgewählt. Wird null zurückgegeben, bleibt die vorhandene Selektion beim Nutzer bestehen.

Beispiel:

```
function get_selected() : ?array
{
  if ($this->show_my_selection) {
    return array(64,236,6,353);
  }

  return null;
}
```

1.4.5. has_multi_selection

Syntax:

bool has_multi_selection()

Beschreibung:

(Ab v4.10) Bestimmt, ob eine Mehrfachauswahl im Grid möglich ist. Wird die Methode nicht überschrieben, ist die Mehrfachauswahl aktiv.

1.4.6. get_tooltip

Syntax:

array get_tooltip($params)

Beschreibung:

Die Methode wird aufgerufen, wenn durch den Nutzer ein Tooltip angefordert wird. Dies geschieht, wenn dieser sich mit dem Cursor auf einer Zelle im Grid , auf einer Zelle im Header oder auf einer Gruppenzeile befindet. Der Parameter *$params* ist ein assoziatives Array. Der wichtigste Parameter ist "type", dieser kann *cell* für Zellen, *header* für die Kopfzeile oder *groupline* für die Gruppenzeile sein.

Abhängig vom Typen werden die folgenden Parameter mitgeliefert:

- *cell*
  - *column_id* - Id der Spalte, welche in get_columns() als *field* definiert wurde.
  - *item_id* - Id der Zeile, welche in get_data() als Schlüssel definiert wurde.
- *header*
  - *column_id* - Id der Spalte, welche in get_columns() als *field* definiert wurde.
- *groupline*
  - *column_id* - Id der Spalte, welche in get_columns() als *field* definiert wurde.
  - *value* - Der Wert der Gruppe, über welcher sich der Cursor befindet.

Beispiel:

```
function get_tooltip($params)
{
  switch($params["type"])
  {
    case "cell":
      return $myObject->getTooltip($params["column_id"], $params["item_id"]);
    case "header":
      return $myObject->getHeaderTooltip($params["column_id"]);
    case "groupline":
      return $myObject->getGrouplineTooltip($params["column_id"], $params["value"]);
  }

  return null;
}
```

1.4.7. get_name

Syntax:

string get_name()

Beschreibung:

Die Methode wird aufgerufen, wenn ein Export durch den Nutzer gestartet wird. Sie gibt den Namen der Datei zurück.

Beispiel:

```
function get_name() : string
{
  return 'mein_grid';
}
```

2. Vererbte Methoden

- addon_base::init()
- addon_base::add_prefix($word)
- addon_base::get_option($option)
- addon_base::get_lang($langvar)