Simple Groupware sgsML Reference Guide

sgsML enables quick customization and creation of powerful web applications

(February 2012)

Simple Groupware sgsML Reference Guide

Table of contents

Links

- Download
- User Manual
- Forum (Google Groups)
- Contact
- Support Request
- Feature Request
- Submit a patch

sgsML Reference Guide

<< contents

This guide describes all aspects of sgsML. Starting from tags and attributes to custom trigger functions, modifiers, formatters and validators.

sgsML is the abbreviation for Simple Groupware Solutions Markup Language, and represents a programming language that enables quick customization and creation of powerful web applications.

A small introduction to sgsML can be found here. The FAQs for sgsML are collected here.

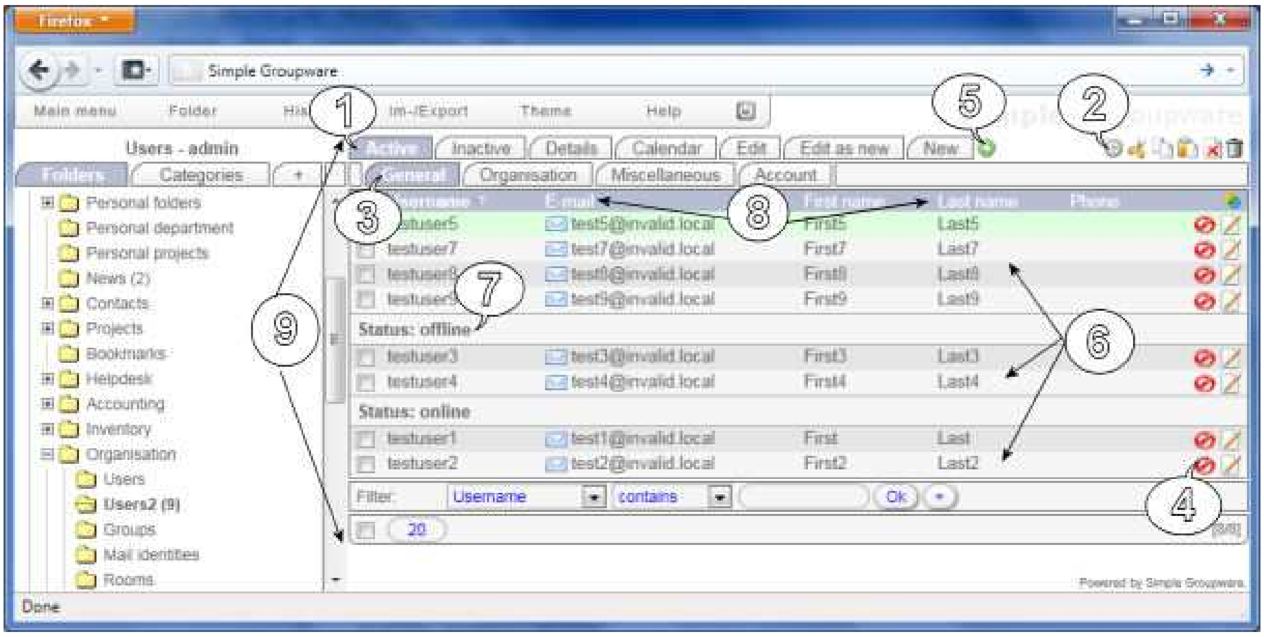
This page contains all basic attributes and tags. All extended attributes are described here.

Note: sgsML always needs to be valid XML

Beginning with release 0.720, fields can be created and changed directly in the GUI. Simply log in as the super administrator and navigate to "/Workspace/System/Customize/Fields".

sgsML elements

1) Views: containing New, Edit, Edit as new
2) View buttons: containing Cut-Copy-Paste, Delete, Purge
3) Tabs, 4) Single buttons, 5) Quick add, 6) Assets, 7) Groups, 8) Fields, 9) Module



Username	E-mail	First name	Last name	Phone
testuser5	test5@invalid.local	First5	Last5	
testuser7	test7@invalid.local	First7	Last7	
testuser8	test8@invalid.local	First8	Last8	
testuser9	test9@invalid.local	First9	Last9	
Status: offline				
testuser3	test3@invalid.local	First3	Last3	
testuser4	test4@invalid.local	First4	Last4	
Status: online				
testuser1	test1@invalid.local	First	Last	
testuser2	test2@invalid.local	First2	Last2	

sgsML elements

Tag structure

- <table>
  - <view>
  - <tab>
  - <viewbutton> (optional)
  - <singlebutton> (optional)

- <field>
  - <data> (optional)
  - <filter> (optional)
  - <validate> (optional)
  - <link>, <linktext> (optional)
  - <notin>, <onlyin> (optional)
  - <description> (optional)

Other topics

- Automatic system fields
- PHP method signatures
- Templates
- Module naming
- Extensions
- Examples
- Release changes

<table> tag

Table is the first tag in the structure and contains general information about the module. All other tags are nested under the "table" tag.

Example:

<table modulename="{t}Tasks{/t}" default_view="display" orderby="ending" order="asc" limit="20" enable_new="true" enable_edit="true" enable_delete="true" enable_empty="true">

Basic attributes:

- **modulename**: Name of the module, displayed in the GUI.
  required: yes
  translatable: yes
  example: modulename="{t}Contacts{/t}" or modulename="Contacts"
  used in: contacts.xml (and more)
- **default_view**: Name of the default view shown when opening the folder. If undefined, first view will be used.
  example: default_view="display"
  used in: contacts.xml (and more)
- **orderby**: Name of the field that sets the default sorting of the assets. If undefined, assets are unsorted by default.
  example: orderby="lastname"
  used in: contacts.xml (and more)
- **order**: Defines the order (ascending or descending) used in the default sorting of the assets. If undefined, ascending will be used.
  possible values: asc, desc
  example: order="asc"
  used in: contacts.xml (and more)
- **limit**: Defines the maximum number of assets to display on a page. If undefined, 20 will be used.
  example: limit="30"
  used in: contacts.xml (and more)

- **groupby**: Defines a field name which is used to group the assets. If undefined, assets are not grouped.
  example: groupby="category"
  used in: search.xml
- **group**: Defines the order (ascending or descending) used to sort the groups of assets. If undefined, group names will be sorted ascending.
  possible values: asc, desc
  example: group="asc"
  used in: ---
- **enable_new**: Adds "new" and "edit_as_new" views and a "paste" button to the module. If undefined, creating new assets will not be possible.
  example: enable_new="true"
  used in: contacts.xml (and more)
- **enable_new_only**: Adds a "new" view and a "paste" button to the module. If undefined, creating new assets will not be possible.
  example: enable_new_only="true"
  used in: nodb_backups.xml, nodb_fs.xml (and more)
- **enable_edit**: Adds an "edit" view to the module. If undefined, editing assets will not be possible.
  example: enable_edit="true"
  used in: contacts.xml (and more)
- **enable_delete**: Adds buttons "delete" (with trash) and "cut" to the module. If undefined, it is not possible to delete or cut/paste assets.
  example: enable_delete="true"
  used in: contacts.xml (and more)
- **enable_purge**: Adds buttons "delete" (without trash) and "cut" to the module. If undefined, it is not possible to delete or cut/paste assets.
  example: enable_purge="true"
  used in: events.xml, session.xml (and more)
- **enable_empty**: Adds "empty folder" (with trash) button to the module. If undefined, it is not possible to delete all assets in a folder at once.
  example: enable_empty="true"
  used in: contacts.xml (and more)
- **enable_purgeall**: Adds "empty folder" (without trash) button to the module. If undefined, it is not possible to delete all assets in a folder at once.
  example: enable_purgeall="true"
  used in: events.xml, session.xml (and more)
- **where**: Adds an additional where condition to all select queries. If undefined, only default where statements are added (e.g. "folder in (@folders@)", "id in (@item@)")
  ---
  example: where="created < DATE_SUB(NOW(), INTERVAL 1 HOUR)" (MySQL)
  example #2: where="(private='0' or createdby=@username@)"
  used in: private_items.diff.xml (and more)
- **disable_quick_add**: Disables the "quick add" button at the end of the views. If undefined, new assets can be created with a single line of comma-separated values, representing the required fields and values.
  useful for: too many required fields, values need to match complex select lookups
  example: disable_quick_add="true"
  used in: contactactivities.xml, expenses.xml (and more)
- **quick_add**: Comma separated list of field names to require in the quick add function. If undefined, all fields with the "required" attrbute set to "true" will be used.
  example: quick_add="pagename,title,data"
  used in: cms.xml, contacts.xml (and more)

<view> tag

View is the second tag in the structure and contains information about views in the module. Views in sgsML are similar to views in databases (groups of fields with a special where condition).

Example:

<view name="display" displayname="{t}Open{/t}" groupby="category" where="closed=0" />

Basic attributes:

- **name**: Defines an alpha-numeric identifier of a view.
  required: yes
  example: name="display"
  used in: contacts.xml (and more)
- **displayname**: Name of the view, displayed in the GUI. If undefined, the "name" attribute will be used instead.
  translatable: yes
  example: displayname="{t}Display{/t}"
  used in: contacts.xml (and more)
- **orderby**: Name of the field that sets the default sorting in the current view of assets. If undefined, assets are sorted by the "orderby" attribute in the "table" tag (or unsorted if not present).
  example: orderby="ending"
  used in: helpdesk.xml
- **order**: Defines the order (ascending or descending) used in the default sorting in the current view of assets. If undefined, assets are sorted by the "order" attribute in the "table" tag (or ascending if not present).
  possible values: asc, desc
  example: order="desc"
  used in: helpdesk.xml
- **limit**: Defines the maximum number of assets to display on a page in the current view. If undefined, the "limit" attribute in the "table" tag will be used (or 20 if not present).
  example: limit="20"
  used in: cms.xml (and more)
- **groupby**: Defines a field name which is used to group the assets in the current view. If undefined, assets are grouped by the "groupby" attribute in the "table" tag (or ungrouped if not present).
  example: groupby="category"
  used in: bookmarks.xml, contacts.xml (and more)
- **group**: Defines the order (ascending or descending) used to sort the groups of assets in the current view. If undefined, group names will be sorted by the "group" attribute in the "table" tag (or ascending if not present).
  possible values: asc, desc
  example: group="asc"
  used in: ---
- **where**: Adds an additional where condition to all select queries to the current view. Where conditions are merged with the "where" attribute in the "table" tag. If undefined, only default where statements are added (e.g. "folder in (@folders@)", "id in (@item@)")
  ---
  example: where="activated=1"
  example #2: where="closed=0"
  used in: tasks.xml, helpdesk.xml, events.xml (and more)

- **showinsingleview**: Shows a link to the current view for each asset. Recommended for "Details" or "Edit". If undefined, views will be only visible in the views bar.
  example: showinsingleview="true"
  used in: bookmarks.xml, notes.xml (and more)
- **show_preview**: Shows previews of images and other files. If undefined, only filenames, sizes and modification dates are shown in the view.
  example: show_preview="true"
  used in: contacts.xml, tasks.xml (and more)
- **image_width**, **image_height**: Shows image thumbnails or other previews with a maximum width or height in pixels. If undefined or "show_preview" is not set for the current view, thumbnails will not be resized.
  example: image_width="250" image_height="200"
  used in: gallery.xml, notes.xml (and more)
- **template_mode**: Customizes the details template (templates/asset_details.tpl). The value "noheader" shows the template without field names on the left side. The value "small" shows only fields with non-empty values. And "flat" shows fields with non-empty values, and names in a separate row. If undefined, field names are on the left, values on the right.
  valid values: small, flat, noheader
  example: template_mode="flat"
  used in: notes.xml, faq.xml (and more)
- **tfield_1**, **tfield_2**: Defines two fields shown in the headline of an asset in the details template (templates/asset_details.tpl) or a calendar view (day.tpl, month.tpl, gantt.tpl). The fields are identified by the "name" attribute". If undefined, only the first field of the view with required="true" will be shown in the headline.
  example: tfield_1="firstname" tfield_2="lastname"
  used in: contacts.xml, tasks.xml (and more)
- **showonly**: Shows only a selection of fields in the current view. If undefined, all fields of a module are visible in a view. The "fields" attribute contains field names, separated by "|".
  example: showonly="subject|headers"
  used in: emails.xml, users.xml
- **enable_asset_rights**: Enables individual permissions for every asset. The value "full" creates a new tab for read and write permissions (users and groups) with the default set to "anonymous". The value "owner_read" creates a new tab for read and write permissions with the default set to the current username. And the value "owner_write" creates a new tab for write permissions with the default set to the current username. If undefined, only folder permissions are used to read and write assets.
  valid values: full, owner_read, owner_write
  example: enable_asset_rights="full"
  used in: files.xml, forum.xml, timesheets.xml (and more)
- **changeseen**: Enables seen and unseen states for each asset. Default state is unseen (marked with bold characters). The asset gets marked as seen when the first user opens it in a view which has the "changeseen" attribute set.
  example: changeseen="true"
  used in: emails.xml, nodb_imap.xml (and more)

<tab> tag

Tab is the third tag in the structure and contains groups of fields in the module. Tabs in sgsML are similar to tabs in browsers (windows/views contain tabs). Fields are assigned to tabs with the "simple_tab" attribute. Fields without a "simple_tab" attribute are assigned to the first tab. Tabs can be hidden with the "hide_tabs" or "disable_tabs" attributes.

Example:

```
<tab name="general" displayname="{t}General{/t}" />
```

Attributes:

- **name**: Defines an alpha-numeric identifier of a tab.
  required: yes
  example: name="general"
  used in: contacts.xml, tasks.xml (and more)
- **displayname**: Name of the tab, displayed in the GUI. If undefined, the "name" attribute will be used instead.
  example: displayname="{t}General{/t}"
  used in: contacts.xml, tasks.xml (and more)

<viewbutton>, <singlebutton> tag (optional)

Viewbutton and **singlebuttons** are optional tags in the structure and define additional buttons in a module. View-buttons will be added to the view bar and can be applied to more than one asset. Single-buttons will be added to each asset and only affect one asset.
The "name" attribute serves as a unique identifier.
The "displayname" attribute is the name of the button displayed in the GUI.
The "views" attribute contains view names to apply the filter to, separated by "|".
The "onclick" attribute describes the Javascript function to execute when the button is clicked.
The "icon" attribute can be used to set an icon for the link (located under "ext/icons/").
The "right" attribute sets the permissions required to show the button ("read", "write", "admin").

The "condition" attribute sets the condition(s) required to show a single-button. Operators can be "eq" (equal),"neq" (not equal),"lt" (less than),"gt" (greater than),"like","nlike" (not like),"starts" (starts with) and "oneof" (one of).
Syntax: field|op|value[||field2|op2|value].
Example: condition="status|oneof|ok,fail||closed|neq|0
Example #2: condition="completed|eq|1||status|oneof|closed,canceled"

useful for: custom actions
used in: calendar.xml, tasks.xml (and more)

Examples:

<viewbutton name="custom1" displayname="{t}Custom action{/t}" onclick="ajax('myclass::ajax_myfunc', [tfolder, tview, asset_get_selected_items(true)], locate_folder);" right="write" />

<singlebutton views="display|calendar" name="close" displayname="{t}Close{/t}" onclick="asset_update({closed:'1'},'@id@');" condition="closed|neq|1" right="write" icon="accept.gif" />

<singlebutton views="display" name="take_over" displayname="{t}Take over{/t}" onclick="asset_update({responsibles:sys.username},'@id@');" right="write" icon="user_add.gif" />

<field> tag

Field is the fourth tag in the structure and contains information about fields in the module. Fields in sgsML are similar to fields in a database table (including data types, length and constraints).

Example:

```
<field name="email" displayname="{t}E-mail{/t}" simple_type="text">
  <validate function="email"/>
  <link value="@ext/norefer.php?url=@email@" icon="link_mail.gif"/>
</field>
```

Basic attributes:

- **name**: Defines an alpha-numeric identifier of a field.
  required: yes
  example: name="lastname"
  used in: contacts.xml (and more)
- **displayname**: Name of the field, displayed in the GUI. If undefined, the "name" attribute will be used instead.
  example: displayname="{t}Last name{/t}"
  used in: contacts.xml (and more)
- **simple_type**: Type of the field, used to present a field in views and forms.
  example: simple_type="text"
  used in: contacts.xml, tasks.xml (and more)
  possible types:
  - text: normal text input (one line)
  - textarea: text input (multiple lines)
  - checkbox: input only yes or no
  - files: upload files
  - date: select a date
  - time: time input
  - datetime: input date and time
  - select: select box (needs <data> tag)
  - dateselect: select several dates
  - rating: choose a rating with stars (added in 0.662)
  - int: numeric input
  - float: float numbers
  - pmwikiarea: text input, PmWiki markup
  - wikiarea: text input, text_wiki markup
  - htmlarea: HTML input
  - spreadsheet: spreadsheet component
  - graphviz: text input (Graphviz markup)
  - codearea: source code widget
  - id: unique identifier
  - password: password input
- **required**: Requires a non-empty input value when creating or editing an asset. If undefined, the input value can be empty.
  example: required="true"
  used in: contacts.xml (and more)
- **is_unique**: Requires the field to have a unique value. If undefined, the input value of the current field can exist more than once in assets belonging to the same module.
  example: is_unique="true"
  used in: contacts.xml (and more)
- **sum**, **avg**: Sums up all values or builds the average of the current field at the bottom of the page. If undefined, there will be no additional line at the bottom displaying sums or

averages.
example: sum="true", average="true"
used in: expenses.xml, projects.xml (and more)

- **simple_default**: Defines a default input value used when creating a new asset. If undefined, the input value will be empty by default.
  example: simple_default="1"
  used in: tasks.xml, calendar.xml (and more)
- **simple_tab**: Defines a name of a tab where the field is placed in. If undefined, the field will be placed in the "general" tab.
  example: simple_tab="address"
  used in: contacts.xml, tasks.xml (and more)
- **simple_size**:
  For simple type "select" and "dateselect", it defines the height of the select box (number of lines). If the value is greater than 1, multiple elements can be selected. If undefined, the default height is 3 lines.
  For simple type "rating", it defines the maximum number of stars in the rating field. If undefined, the maximum is 5.
  For simple type "files", it defines the maximum number of files that can be uploaded to the field in one asset. If undefined, an unlimited number of files can be uploaded.
  example: simple_size="1"
  used in: contacts.xml, tasks.xml (and more)
- **simple_file_size**: Sets the maximum file size in bytes for a "files" field. Values can be given in units "M" (megabytes) and "K" (kilobytes). If undefined, files can be uploaded to a maximum configured in "php.ini". Relevant parameters in "php.ini" are "post_max_size" and "upload_max_filesize".
  example: simple_type="files" simple_file_size="5M"
  used in: contacts.xml, tasks.xml (and more)
- **separator**: Defines a separator used to display fields containing multiple values (simple type files, select, dateselect and multitext). If undefined, the separator will be set to ", ". Beginning with version 0.645, the default is "\n".
  example: separator="\n"
  used in: contacts.xml, tasks.xml (and more)
- **nowrap**: Disables automatic text wrapping. If undefined, long texts will be automatically wrapped to the next line.
  useful for: date/time fields
  example: nowrap="true"
  used in: tasks.xml, calendar.xml (and more)
- **allow_custom**: Applies to simple type "select". Allows the user to enter custom values not offered by the select box.
  example: allow_custom="true"
  used in: contacts.xml, tasks.xml (and more)
- **width**: Sets the width of a field in the view. If undefined, all fields will be distributed on the table automatically by the browser. The minimum width is set to 40 pixels per field. Values can be set in "px", "em", "%".
  example: width="50%"
  used in: bookmarks.xml, events.xml (and more)
- **height**: Sets the height of a field in the view. If undefined, the height will be defined by the content of the field. Values can be set in "px", "em", "%".
  example: height="300px"
  used in: notes.xml, gallery.xml (and more)

- **<link>**, **<linktext>**: Adds a link to the current field. The link can be placed before the current value (<link> tag) or directly in the current value (<linktext> tag). Variables like @folder@, @id@ or @field_name@ automatically get replaced with their current values. The link can be added to all views or some views specified in the "views" attribute, separated by "|". When the "value" attribute starts with a "#", the link is opened in the horizontal preview pane. When the "value" attribute starts with a "%", the link is opened in the vertical preview pane on the right side. When the "value" attribute starts with a "@", the link is opened in a new browser window. The "icon" attribute can be used to set an icon for the link (located under "ext/icons/"). URL parameters are described here.
  ---
  example: <link value="@ext/norefer.php?url=@url@"/>
  example #2: <linktext views="display" value="#index.php?folder=@folder@&view=details&iframe=1&item=@id@"/>
  example #3: <link value="skype:@skype@?userinfo" icon="phone.gif"/>
  used in: contacts.xml, tasks.xml

- **<filter>**: Calls a PHP method to modify the output of a field in a view. There can be multiple <filter> tags inside a <field> tag. Method parameters can be added, separated by "|". Default class is "modify", can be changed with prefix "classname::". Filters can be applied to all views or some views specified in the "views" attribute, separated by "|". The "type" attribute (optional) can be set to "_fgstyle" to set a foreground CSS attribute or "_bgstyle" to set a background CSS attribute.
  ---
  translatable: yes
  useful for: modifying output of a field
  example: <filter views="all" function="shortdatetimeformat"/>
  example #2: <filter views="all" function="percent"/>
  example #3: <filter views="display" function="shortmessage|100"/>
  example #4: <filter views="all" function="dateformat||{t}m/d/Y{/t}"/>
  example #5: <filter views="display|details" function="myclass::validate_url"/>
  used in: contacts.xml, tasks.xml (and more)
  linkselect: Appends the value of a field with a couple of links pointing to other records in the database. Links are separated with a <separator> string. Two fields are selected from a table: The first field is used as the ID of the record to be linked. The second field serves as the text for the link. Variables like @field_name@ automatically get replaced with their current values in the where field. The maximum number of links is 100 by default and can be changed with the <limit> parameter.
  ---
  syntax: <filter views="<views>" function="linkselect|<separator>|<table>|<key-column>,<value-column>|<where>|<order-by>|<limit>"/>
  example: <filter views="all" function="linkselect|, |simple_intranet|id,pagename|folder=@folder@ and mainpage='0'|pagename"/>
  showselect: Similar to linkselect, but it only presents values without links.
  ---
  syntax: <filter views="<views>" function="showselect|<separator>|<table>|<key-column>,<value-column>|<where>|<order-by>|<limit>"/>
  example: <filter views="all" function="showselect|\n|simple_tasks|'',concat('Average: ';round(avg(progress)*100);'%')|project=@projectname@|"/>

- **<data>**: Defines the list of options used in simple type "select". There can be multiple <data> tags inside a <field> tag. Can be a list of values, separated by "|" (attribute "values"). Can also be a list of key-value pairs, separated by "_##_" and "|" (see example #2). The

"reverse" attribute can be used to translate keys in normal views to values.
Can be a PHP method which returns the list. Method parameters can be added, separated by "|". Default class is "select", can be changed with prefix "classname::".
The "title" attribute is optional and can be used to describe multiple <data> blocks in the form.
---
translatable: yes
used in: contacts.xml, tasks.xml (and more)
examples:
<data title="Companies" values="low|normal|urgent"/>
<data reverse="true" values="0_##_{t}low{/t}|1_##_{t}normal{/t}|2_##_{t}urgent{/t}"/>
<data function="dbselect|simple_companies|id,companyname||companyname asc|10"/>
<data
function="dbselect|<table>|<key-column>,<value-column>|<where>|<order-by>|<limit>|<no_permisions>"/>
<data function="dbselect|(select a.* from table1 a, table2 b where a.id=b.id)|<key-column>,<value-column>||<order-by>|<limit>"/>
<data function="dbselect|simple_sys_users|username,concat(lastname;’; ’;firstname)||lastname asc|10"/>
<data function="myclass::mymethod|params..."/>
Note: For the "dbselect" function, all permissions are checked automatically. Datasets from external tables not related to Simple Groupware will only show up if the 6th parameter is set to "no_permissions". Permissions are checked against the table "simple_sys_table".
Note: To use the "concat" function (SQL) in <value-column>, replace "," with ";", e.g. concat(lastname;’ ’;firstname)
Note: When there a more entries than <limit>, the result is empty and the user is forced to use the search function. The <limit> parameter is used as a maximum amount of entries to fetch.

- **<validate>**: Calls a PHP method to validate the current field before saving it to the database. There can be multiple <validate> tags inside a <field> tag. Method parameters can be added, separated by "|". Default class is "validate", can be changed with prefix "classname::".
---
translatable: yes
useful for: validating user input
example: <validate function="numeric"/>
example #2: <validate function="email"/>
example #3: <validate function="url"/>
example #4: <validate function="regexp|/^[0-9]+$/|ID must be numeric"/>
example #5: <validate function="myclass::validate_url" />
used in: contact.xml, tasks.xml (and more)

- **<notin>**, **<onlyin>**: Not-in removes the current field from some views. Only-in shows the current field only in some views. If undefined, all fields of a module are visible in a view. The "views" attribute contains view names, separated by "|".
---
useful for: remove fields from views
example: <notin views="display"/>, <onlyin views="details|calendar"/>
used in: contacts.xml, tasks.xml (and more)

- **<description>**: Adds a description to a field in "new", "edit" and "edit as new" views. The attribute "title" is the text of the link. The attribute "hint" is the tooltip shown for the link. The attribute "value" is the Javascript code which gets executed when the link is clicked. The attributes "title" and "hint" are optional.
example: <description title="Help" hint="Click here for help." value="alert(’minimum = 15 secs’);"/>

used in: contact.xml, tasks.xml (and more)

Note: Custom classes can be put under "custom/core/classes/" (will be tried first) or "ext/core/classes/" (reserved for extensions). E.g. "class mymethods {}" in "custom/core/classes/mymethods.php".

Note: Showing a field in a view is evaluated by the following order: notinall (attribute in <field>), notin (tag in <field>), onlyin (tag in <field>), showonly (attribute in <view>)

Note: In Simple Groupware, translatable strings are marked with "{t}", "{/t}". For example "{t}Open{/t}" will be translated to "Offen" in German.

Automatic system fields (hidden by default)

- **id**: Numeric unique identifier for an asset (primary key). Consists of asset ID and server ID (2 decimals), e.g. 201 for asset ID 2 and server 01.
- **folder**: ID of the folder that contains the asset (reference to simple_sys_tree).
- **created**: Unix timestamp pointing to the time when the asset was created.
- **lastmodified**: Unix timestamp pointing to the time when the asset was last modified.
- **createdby**: Username of the user who created the asset.
- **lastmodifiedby**: Username of the user who did the last modification.
- **dsize**: Numeric value indicating the size (bytes) of the files attached to the asset.
- **history**: Text field to track information about who changed what data and which fields of an asset.

PHP method signatures

- **Filter method**
  Example: <filter views="display" function="myclass::my_filter_method|100|abc"/>
  Signature: static function my_filter_method($value, $params, $row)
  Parameters:
  $value: Value to filter (string)
  $params: input parameters, array(100, 'abc')
  $row: current asset, array(field_name => array('data'=>array(...), 'filter'=>array(...)), ...)
  [data = unfiltered values, filter = filtered values]
  Return: string (filtered value)

- **Validate method**
  Example: <validate function="myclass::my_validate_method|100|abc"/>
  Signature: static function my_validate_method($value, $params, $field)
  Parameters:
  $value: Value to validate (string)
  $params: input parameters, array(100, 'abc')
  $field: current sgsML field, array("NAME"=>"...", "DISPLAYNAME"=>"...", ...)
  Return: string empty (valid value) or error message (invalid value)

- **Data method**
  Example: <data function="myclass::my_data_method|100|abc"/>
  Signature: static function my_data_method($params)
  Parameters:
  $params: input parameters, array(100, 'abc')
  Return: array(Key => Value, Key2 => Value2)
  [User sees values in the form, keys will be stored in the database]

Templates

The following templates are included:

- **admin**: Used to display the admin overview page under "/Workspace/System".
  location: templates/asset_admin.tpl
  additional attributes: none
  used in: nodb_admin.tpl
- **display**: Used to display assets with one line per asset.
  location: templates/asset_admin.tpl
  additional attributes: none
  used in: contacts.xml, tasks.xml (and more)
- **details**: Used to display assets with one line for every field.
  location: templates/asset_admin.tpl
  additional attributes: tfield_1, tfield_2, template_mode ("view" tag)
  used in: contacts.xml, tasks.xml (and more)
- **edit**: Form used to edit and create assets with one block for every asset.
  location: templates/asset_edit.tpl
  additional attributes: none
  used in: sgsml.php (attributes: enable_new, enable_edit, enable_edit_as_new)
- **free**: Used to display assets in rows and columns with one block for every asset and several blocks per row. Number of blocks per row defined in "cols" attribute.
  location: templates/asset_free.tpl
  additional attributes: cols, row_height ("view" tag)
- **portal**: Used to display URLs in iframes with one block for every asset and several blocks per row. Number of blocks per row defined in "cols" attribute.
  location: templates/asset_portal.tpl
  notes: does not support tabs
  additional attributes: cols ("view" tag)

CMS templates: Used in the CMS (cms.php) to display pages. Templates can be selected for every asset in the CMS module.

- **pmwiki.tpl**: Default template used to display a page.
  location: cms/pmwiki.tpl
- **pmwiki_rtl.tpl**: Default template with right-to-left layout.
  location: cms/pmwiki_rtl.tpl
- **rss.tpl**: Special template used to display a RSS feed of the latest pages updated. Includes only pages with the flag "Include in RSS feed" set.
  location: cms/rss.tpl
  URL: cms.php?rss
- **sitemap.tpl**: Special template used to display a XML stream of the latest pages updated. Includes all pages.
  location: cms/sitemap.tpl
  URL: cms.php?sitemap

Module naming

All modules are located at "modules/schema*". The folder "schema" contains modules for all users. The folder "schema_sys" contains all administrative modules. All modules are named with "<module_name>.xml". Filenames can contain a "nodb_" prefix to indicate that the database schema will not be processed, meaning NO automatic creation of tables and fields.

Extensions

sgsML can be also used in extensions. For getting started, it is recommended to take a look at the example extension which can be downloaded here.

Examples

Here are some example modules:

- contacts.xml
- tasks.xml
- bookmarks.xml

Without translation:

- contacts_en.xml
- tasks_en.xml
- bookmarks_en.xml

Release changes

- v0.732: added tfield_1, tfield_2 for calendar views
- v0.722: added vertical preview pane for the "<link" and "<linktext"
- v0.707: added "no_permissions" parameter for the "dbselect" function
- v0.701: added "condition" attribute to "singlebutton" tag
- v0.662: added "reverse" attribute to "data" tag
- v0.662: added simple type "rating"
- v0.646: renamed "nosql_" prefix for filenames to "nodb_"
- v0.645: renamed "multiple" attribute to "separator"
  ("multiple" will continue to work)
- v0.645: changed default value for "multiple" attribute to "\n"

sgsML Extended Attributes

<< contents

This page contains all extended attributes and tags for sgsML. All basic attributes are described here.

Note: All extended attributes and tags are optional.

Tag structure

- <table>
  - <view>
  - <tab> (basic tag)
  - <viewbutton> (optional, basic tag)
  - <singlebutton> (optional, basic tag)
  - <rowfilter> (optional)
  - <rowvalidate> (optional)
  - <field>
    - <data> (optional, basic tag)
    - <filter> (optional, basic tag)
    - <validate> (optional, basic tag)
    - <store>, <restore> (optional)
    - <link>, <linktext> (optional, basic tags)
    - <notin>, <onlyin> (optional)
    - <readonlyin>, <hiddenin> (optional)
    - <description> (optional, basic tag)

<table> tag: extended attributes

- **load_library**: Loads an additional library in the bootstrap. The base path is "<sgs-dir>/bin/" or "<sgs-dir>/src/".
  useful for: modules that need a PHP library
  example: load_library="lib/pmwiki/pmwiki.php"
  used in: cms.xml, nodb_pmwiki.xml
- **load_css**: Loads an additional stylesheet in the head section of the page. The base path is "<sgs-dir>/bin/" or "<sgs-dir>/src/".
  useful for: modules that need a custom stylesheet
  example: load_css="ext/cms/styles.css"
  used in: cms.xml, nodb_pmwiki.xml
  available since: 0.643
- **name**: Defines a table name which overrides the normal table name in all queries. If undefined, the module name will be used as the table name.
  useful for : overwriting the default table name
  example: name="simple_sys_tree"
  used in: nodb_rights_edit.xml
- **custom_name**: Defines a table name or a sub-query which overrides the normal table name in all "select" queries. If undefined, the module name will be used as the table name.
  useful for: map data into another (sub-)schema
  example: custom_name="simple_contacts_archive"
  used in: search.xml, nodb_calendar_contacts.xml

- **default_sql**: Defines a default query used to retrieve assets. If undefined, a select query is automatically built by the "fields" and "where" statements assigned to a view. Variables like @folder@ or @table@ automatically get replaced with their current values.
  By default, only assets of the current folder are shown in the current view. So if a "default_sql" query returns a "folder" field with a different ID, assets are not shown because the access is denied. You can bypass the folder check by adding the attribute "nosqlfolder" with the value "true".
  ---
  special values: no_select (don't run selects)
  example: default_sql="show full columns from @table@" (MySQL)
  used in: nodb_structure.xml, nodb_admin.xml (and more)
- **template**: Defines the default Smarty template, used to render the views. If undefined, each view needs to define it's own template attribute or uses the view's name to set a template implicitly.
  useful for: refer to a different template without changing the view name
  example: template="display" (for templates/asset_display.tpl)
  used in: nodb_schema.xml, nodb_rights.xml (and more)
- **no_search_index**: Disables the search indexer for the module. If undefined, all new or changed assets will be put into the search index. Modules that contain "_nodb_" in the filename are automatically excluded.
  example: no_search_index="true"
  used in: search.xml, session.xml (and more)
- **disable_***: Disables automatic views, buttons or fields in the module. If undefined, some special views, buttons and fields are automatically added to the module.
  Notification and summary allow people to get notified about changes.
  Bgcolor allows to mark assets with a background color as label.
  attribute names for views: disable_{history|index|schema|search|copy|structure|rights}
  attribute names for buttons: disable_{paste|copy|cut}
  attribute names for fields: disable_{bgcolor|notification}
  example: disable_history="true"
  used in: session.xml
- **enable_calendar**: Enables the calendar box above the tree. Contains relevant fields for time frame selection, can be only field (created), two fields (begin, end) or nine fields (including recurrence, intervals and exclusions). If undefined, only events that match the selected time frame will be shown.
  example: enable_calendar="created"
  example #2: enable_calendar="begin,ending"
  used in: calendar.xml, events.xml (and more)
- **hide_calendar**: Hides the calendar pane. If undefined, all entries with begin and end date will be displayed in the calendar pane.
  example: hide_calendar="true"
  used in: ---
- **sql_handler**: Defines a class used to delegate all queries to. The prefix of the class name "lib_" is left out. The folder for a library of a module is "modules/lib/". Is mainly used for external data structures defined over mountpoints (see data handlers and "modules/lib/default.php").
  ---
  example: sql_handler="fs" (class name lib_fs)
  example classes: modules/lib/fs.php, modules/lib/csv_contacts.php
  interface definition: modules/lib/default.php
  used in: nodb_fs.xml, nodb_csv_contacts.xml (and more)
- **disable_rights**: Disables folder permissions for a module. If undefined, folder permissions can be set for every folder of the module.
  example: disable_rights="true"
  used in: nodb_fs.xml, nodb_imap.xml (and more)

- **disable_folder_operations**: Disables folder operations for a module. If undefined, folders can be created, renamed and deleted.
  example: disable_folder_operations="true"
  used in: nodb_pop3.xml, nodb_xml.xml (and more)
- **trigger_new**: Calls a PHP method after a new asset was created. Can be more than one method, separated by "|". Method parameters can be added, separated by ":". Default class is "trigger", can be changed with prefix "classname::".
  example: trigger_new="createcontact"
  example #2: trigger_new="somemethod|other_class::anothermethod:100:abc"
  example #3: trigger_new="runxml:modules/coreprojects.xml:projects"
  ---
  signature: static function anothermethod($id, $row, $params, $table)
  [$id = asset id, $row = dataset array, $params = parameters array(100,’abc’), $table = table name, return string empty (success) or error message (trigger failed)]
  ---
  used in: contacts.xml, projects.xml (and more)
- **trigger_edit**: Calls a PHP method after an asset was changed. Can be more than one method, separated by "|". Method parameters can be added, separated by ":". Default class is "trigger", can be changed with prefix "classname::".
  example: trigger_edit="editcontact"
  example #2: trigger_edit="somemethod|other_class::anothermethod:100:abc"
  ---
  signature: static function anothermethod($id, $row, $params, $table)
  [$id = asset id, $row = dataset array, $params = parameter array(100,’abc’), $table = table name, return string empty (success) or error message (trigger failed)]
  ---
  used in: contacts.xml, projects.xml (and more)
- **trigger_delete**: Calls a PHP method after an asset was deleted. Can be more than one method, separated by "|". Method parameter can be added, separated by ":". Default class is "trigger", can be changed with prefix "classname::".
  example: trigger_delete="deletecontact"
  example #2: trigger_delete="somemethod|other_class::anothermethod:100:abc"
  ---
  signature: static function anothermethod($id, $row, $params, $table)
  [$id = asset id, $row = dataset array, $params = parameter array(100,’abc’), $table = table nae, return string empty (success) or error message (trigger failed)]
  ---
  used in: contacts.xml, projects.xml (and more)
- **disable_trigger_ccp**: Disables all triggers on cut-copy-paste. If undefined, the paste command will execute a trigger defined in the "trigger_new" attribute. If undefined, the cut command will execute a trigger defined in the "trigger_delete" attribute.
  example: disable_trigger_ccp="true"
  used in: emails.xml, nodb_imap.xml
  available since: 0.651

Note: Custom classes can be put under "custom/core/classes/" (will be tried first) or "ext/core/classes/" (reserved for extensions). E.g. "class mymethods {}" in "custom/core/classes/mymethods.php".

<view> tag: extended attributes

- **template**: Defines the Smarty template, used to render the current view. If undefined, the "template" attribute in the "table" tag will be used or (if not present), the "name" attribute of the current view will be used to determine the Smarty template (e.g. name="display" => templates/asset_display.tpl).
  useful for: refer to a different template without changing the view name
  example: template="free" (for templates/asset_free.tpl)
  used in: gallery.xml, portal.xml (and more)
- **enable_calendar**: Enables the calendar box above the tree. Contains relevant fields for time frame selection, can be only field (created), two fields (begin, end) or nine fields (including recurrence, intervals and exclusions). If undefined or not present in the "table" tag, only events that match the selected time frame will be shown in the current view.
  ---
  example: enable_calendar="created"
  example #2: enable_calendar="begin,ending"
  used in: calendar.xml, events.xml (and more)
- **hide_calendar**: Hides the calendar pane in the current view. If undefined, the "hide_calendar" attribute in the "table" tab will be used or (if not present), all entries with begin and end date will be displayed in the calendar pane.
  example: hide_calendar="true"
  used in: ---
- **hide_tabs**: Hides some tabs in the current view. If undefined, all tabs are visible. Multiple values can be used, separated by "|".
  example: hide_tabs="tabname1|tabname2"
  used in: ---
- **disable_tabs**: Hides the tabs bar and shows all (visible) fields in one tab.
  example: disable_tabs="true"
  used in: users.xml
- **default_sql**: Defines a default query used to retrieve assets in the current view. If undefined, the "default_sql" attribute in the "table" tag will be used or (if not present), a select query is automatically built by the "fields" and "where" statements assigned to a view. Variables like @folder@ or @table@ automatically get replaced with their current values.
  ---
  special values: no_select (don’t run selects)
  example: default_sql="show full columns from @table@" (MySQL)
  used in: sgsml.php
- **nosqllimit**: Shows all assets in a folder. If undefined, only a maximum number of assets will be displayed on a page, including links to next and previous pages.
  example: nosqllimit="true"
  used in: ---
- **nosqlorder**: Shows assets unsorted. If undefined, assets are displayed sorted by the "orderby" attribute defined in "view" or "table" tags.
  example: nosqlorder="true"
  used in: ---
- **nosqlwhere**: Shows all assets in the view. If undefined, filters from "where" attributes in "view" or "table" tags will be applied to all select queries.
  example: nosqlwhere="true"
  used in: ---
- **nosqlfolder**: Shows assets of all folders in the view. If undefined, only assets of the current folder will be shown in the current view.
  useful for: views not depending on folders
  example: nosqlfolder="true"
  used in: events.xml, tree.xml (and more)
- **visibility**: Defines where the link to the view should be shown. If undefined, the link will be

displayed as a button in the view bar at the top. The value "hidden" will not show a link to the view. The value "bottom" will show a link below the tree at the left side. And "active" will only show a button in the view bar, if the view is currently selected.
valid values: hidden, active, bottom
example: visibility="bottom"
used in: emails.xml, users.xml (and more)

- **noviewbuttons**: Hides buttons in the current view. If undefined, all buttons defined as "view buttons" are displayed in the upper right in the current view. Can hide all buttons with value "true" or some buttons with names in the value, separated by "|".
  example: noviewbuttons="true"
  example #2: noviewbuttons="paste|delete"
  used in: sgsml.php, calendar.xml
- **nosinglebuttons**: Hides buttons for each asset in the current view. If undefined, all buttons defined as "single buttons" are displayed for each asset. Can hide all buttons with value "true" or some buttons with names in the value, separated by "|".
  example: nosinglebuttons="true"
  example #2: nosinglebuttons="take_over|close"
  used in: sgsml.php, calendar.xml (and more)
- **doubleclick**: Defines a name of a view that will be opened when a double-click is done on the background of an asset.
  example: doubleclick="details"
  used in: ---
- **accesskey**: Defines an accesskey which opens the view. If undefined, views can be accessed by numbers, e.g. first view Alt-1, second view Alt-2, etc.
  example: accesskey="e"
  used in: sgsml.php
- **schema_mode**: Defines a mode in which the current view can be used. The value "new" is used for forms that create new assets. The value "edit" is used for forms that change assets. The value "edit_as_new" is used for forms that create new assets from existing assets. The value "static" hides filters and pagers from the view and is used in combination with the "function" attribute. If undefined, creating or editing assets in a view won't be allowed.
  ---
  valid values: edit, create, edit_as_new, static
  example: schema_mode="edit"
  used in: users.xml, emails.xml (and more)
- **schema**: Defines a sub-schema (sgsML) that is used for the current view. If undefined, the sgsML schema from the current module will be used for all views. A sub-schema is the `first` view of a separate module in a separate .xml file. The sub-schema overloads the current view and also attributes from the "table" tag.
  ---
  useful for: map a birthday field in contacts into appointments
  example: schema="sys_nodb_calendar_contacts"
  (modules/schema_sys/nodb_calendar_contacts.xml)
  used in: contacts.xml, users.xml (and more)
- **right**: Defines a permission that is required to access the current view. If undefined, read permissions to the current folder are required.
  valid values: read, write, admin
  example: right="write"
  used in: sgsml.php
- **function**: Calls a PHP method to render the current view. Default class is "custom", can be changed with prefix "classname::". Important feature to integrate custom PHP applications within Simple Groupware.
  example: function="testing"
  example #2: function="other_class::somemethod"

---
signature: static function somemethod($folder, $view)
[$folder = current folder id, $view = current view, return HTML output]
---
used in: nodb_phpinclude.xml

Note: Custom classes can be put under "custom/core/classes/" (will be tried first) or "ext/core/classes/" (reserved for extensions). E.g. "class mymethods {}" in "custom/core/classes/mymethods.php".

<rowfilter> tag

Rowfilter is an optional tag in the structure and contains a PHP method to modify the output of an asset in a view. Compared to the <filter> tag, the result can be applied as a style to the whole asset.
Calls a PHP method to modify the output of an asset in a view. There can be multiple <rowfilter> tags inside a <table> tag. Method parameters can be added, separated by "|". Default class is "modify", can be changed with prefix "classname::". The "name" attribute serves as a unique identifier. The "views" attribute contains view names to apply the filter to, separated by "|". The "type" attribute can be set to "_fgstyle" to set a foreground CSS attribute or "_bgstyle" to set a background CSS attribute. The "views" attribute contains the field names to be validated, separated by "|".

Example:

<rowfilter name="filter1" views="all" type="_bgstyle"
function="isinpast|ending|background-color:#FFDDDD;"/>

<rowfilter name="filter2" views="all" type="_bgstyle" function="myclass::rowfilter|abc|100"/>

Signature: static function rowfilter($row, $params)

- $row = dataset array with "data" and "filter" keys
- $params = parameter array('abc',100)
- return string style attribute(s)

<rowvalidate> tag

Rowvalidate is an optional tag in the structure and contains a PHP method to validate new assets. Compared to the <validate> tag, it validates the whole row in place of one field.
Calls a PHP method to validate an asset before saving it to the database. There can be multiple <rowvalidate> tags inside a <table> tag. Method parameters can be added, separated by "|". Default class is "validate", can be changed with prefix "classname::". The "name" attribute serves as a unique identifier. The "fields" attribute contains the field names to be validated, separated by "|".

Example:

<rowvalidate name="v1" fields="categoryname|cattype"
function="itemsexist|simple_categories|categoryname,cattype"/>

<rowvalidate name="v2" fields="field_a,field_b" function="myclass::rowvalidate|abc|100"/>

Signature: static function rowvalidate($row, $params)

- $row = dataset array
- $params = parameter array('abc',100)
- return string empty (valid asset) or error message (invalid asset)

<field> tag: Extended attributes (optional)

- **simple_default_function**: Defines a default input value by using a PHP method when creating a new asset. If undefined, the input value will be empty by default. Default class is "modify", can be changed with prefix "classname::".
  translatable: yes
  example: simple_default_function="getusername"
  example #2: simple_default_function="dateformat|now|{t}m/d/Y{/t}"
  example #3: simple_default_function="fillform|var1" (request parameter "var1")
  example #4: simpleabc_default_function="myclass::default_method|100|abc"
  ---
  signature: static function default_method($params)
  [$params = parameter array(100,'abc'), return string default value]
  ---
  used in: contacts.xml
- **is_unique_with_trash**: Requires the field to have a unique value and covers also values that exist in the trash folder. If undefined, the input value of the current field can exist more than once in assets belonging to the same module.
  ---
  useful for: unique system values (e.g. usernames)
  example: is_unique_with_trash="true"
  used in: users.xml
  available since: 0.644
- **no_checks**: Avoids checks for insecure HTML. If undefined, all fields will be checked for insecure HTML code (see function modify::htmlfield). This can be useful for filters producing HTML code like forms, that normally get stripped out.
  ---
  useful for: filters producing HTML code like forms (get stripped out by default)
  example: no_checks="true"
  used in: surveys.xml
  available since: 0.643
- **hidden**: Fetches the values for the field from the database, but does not show the field in the view. If undefined, all fields defined in a module will be fetched from the database and shown in the view. Can be overridden by "editable" attriute for "new" and "edit" views.
  ---
  useful for: fields to hide in a view but use them in filters
  example: hidden="true"
  used in: calendar.xml, forum.xml (and more)
- **editable**: Allows to edit a field that is hidden in all views. Used to set a value in "new" and "edit" views, but do not show the value in other views.
  useful for: notifications, assign colors to assets
  example: hidden="true" editable="true"
  used in: tasks.xml, forum.xml (and more)
- **nodb**: Uses a field in a view, but does not create the field in the database table. If undefined, all fields defined in the module will be created in the database.
  example: nodb="true"
  used in: search.xml, stats.xml

- **db_size**: Sets the maximum field size for the field in the database table. If undefined, the field size will be determined by the "simple_type" attribute. Text fields get a limit of 255, IDs get 15, dates get 10 and text areas, select boxes are limited to 64k.
  example: db_size="VALUE"
  used in: stats.xml, tree.xml (and more)
- **db_type**: Sets a field type for the field in the database table. If undefined, the field type will be determined by the "simple_type" attribute. Text fields get be "varchar", text areas and select boxes get "text", dates, bools and IDs are saved as "int".
  ---
  useful for: fields requiring bigger fields or numeric representations
  example: db_type="int"
  used in: calendar.xml
- **no_search_index**: Forbids to put the field values in the search index. If undefined, all field values will be indexed and presented in the search results.
  useful for: password fields
  example: no_search_index="true"
  used in: tree.xml, users.xml (and more)
- **disable_ccp**: Does not copy the field value when using copy-paste. If undefined, all field values will be copied to the new dataset.
  useful for: password fields
  example: disable_ccp="true"
  used in: users.xml, surveys.xml (and more)
- **input_height**: Sets the height of an input field. If undefined, the height will be set to "300" pixels. Only used for simple types "htmlarea", "codearea" and "spreadsheet".
  example: input_height="300"
  used in: htmldocs.xml
- **notinall**: Excludes a field from all views and from the search index. Also avoids fetching the field from the database. Can be overridden for some views with the "onlyin" tag.
  useful for: internal fields like created, lastmodified
  example: notinall="true"
  used in: tasks.xml, projects.xml
- **form**: Sets attributes used to display the current field in a form (edit, new, edit as new). Only defined for simple_type "textarea". "no_template_bar" disables the option to add templates to the text box. "no_formatting_button" hides the button "Text formatting rules".
  valid values: no_formatting_button, no_template_bar
  example: form="no_formatting_button|no_template_bar"
  used in: emails.xml, nodb_imap.xml (and more)
  available since: 0.651

<field> tag: extended sub-tags

- **<KEY>**: Makes the field a primary key in the database table. Primary keys are unique indexes which speed up lookups in the database.
  example: <KEY/>
  used in: hosts.xml, search.xml
- **<INDEX>**: Creates an index over the field in the database table. Indexes speed up lookups in the database.
  example: <INDEX/>
  used in: contacts.xml, tasks.xml (and more)
- **<INDEX_FULLTEXT>**: Creates a fulltext index over the field in the database table. Fulltext indexes speed up search operations on texts.
  example: <INDEX_FULLTEXT/>
  used in: search.xml

- **<store>**: Calls a PHP method to modify the current field before saving it to the database. There can be multiple <store> tags inside a <field> tag. Method parameters can be added, separated by "|". Default class is "modify", can be changed with prefix "classname::".
  ---
  useful for: convert user input before saving, e.g. convert human readable dates to unix timestamps
  example: <store function="html_tidy"/> (reduce HTML code)
  example #2: <store function="datetime_to_int"/>
  example #3: <store function="myclass::store_method|100|abc"/>
  ---
  signature: static function store_method($value, $row, $params)
  [$value = value to be stored, $row = dataset array, $params = parameter array(100,’abc’), return string storable value]
  ---
  used in: contacts.xml, htmldocs.xml (and more)

- **<restore>**: Calls a PHP method to restore the current field before displaying the field in the form of the "edit" and "edit as new" views. There can be multiple <restore> tags inside a <field> tag. Method parameters can be added, separated by "|". Default class is "modify", can be changed with prefix "classname::". The "views" attribute contains view names, separated by "|". Will be applied before "filters" attribute.
  ---
  useful for: make fields empty in "edit as new" view, convert unix timestamps to human readable dates
  example: <restore views="edit_as_new" function="empty_str"/>
  example #2: <restore views="reply|replyall" function="empty_str"/>
  example #3: <restore views="all" function="myclass::restore_method|100|abc"/>
  ---
  signature: static function restore_method($value, $params, $row)
  [$value = value to be restored, $params = parameter array(100,’abc’), $row = dataset array with "data" and "filter" keys, return string restored value]
  ---
  used in: cms.xml, forum.xml (and more)

- **<readonlyin>**: Read-only-in removes the field from "new" and "edit" views. If undefined, all fields of a module are visible in a view. The "views" attribute contains view names, separated by "|".
  useful for: non-editable fields like IDs
  example: <readonlyin views="edit"/>
  used in: emails.xml, users.xml (and more)

- **<hiddenin>**: Hidden-in fetches the current field from the database, but does not show it in some views. If undefined, all fields of a module are editable in "new" and "edit" views. The "views" attribute contains view names, separated by "|".
  ---
  useful for: fields to hide in a view but use them in filters
  example: <hiddenin views="display"/>
  used in: surveys.xml

Note: Custom classes can be put under "custom/core/classes/" (will be tried first) or "ext/core/classes/" (reserved for extensions). E.g. "class mymethods {}" in "custom/core/classes/mymethods.php".

Note: Custom types can be put under "custom/core/types/" (will be tried first) or "ext/core/types/" (reserved for extensions). E.g. "class type_mytext {}" in "custom/core/types/mytext.php".

Note: Showing a field in a view is evaluated by the following order: notinall (attribute in <field>), notin (tag in <field>), onlyin (tag in <field>), showonly (attribute in <view>)

Note: In Simple Groupware, translatable strings are marked with "{t}", "{/t}". For example "{t}Open{/t}" will be translated to "Offen" in German.

Release changes

- v0.730: added "disable_notification" and "disable_bgcolor" attributes to the table tag.
- v0.720: changed form attribute separator from "," to "|".
- v0.661: added "form" attribute to the "field" tag.
- v0.651: added "disable_trigger_ccp" attribute to the "table" tag.
- v0.646: renamed "cust_name" attribute to "custom_name"
  ("cust_name" will continue to work)

sgsML Frequently asked questions / FAQ

<< contents

coming up soon ...