⇐Back to the list of samples

# Struct: complex sample (Aggregation)

** Work in Progress **

Prerequisites:

- installed struct Plugin
- installed Bureaucracy Plugin
- all schemes from Medium Complex Sample

## Introduction

We want to continue to explore the possibilities of Struct and continue to use aggregations to achieve that. Our namespace `infrastructure:systems` houses systems. Let's create an overview and create a page `systems` in the namespace `infrastructure`.

## Create Overview

The following text generates an overview, staggered by our types of systems and with links to the respective pages.

systems.txt

```
====== Systems ======

Overview over the Systems in our environment:

===== Servers =====
---- struct table ----
schema: system
cols: %title%, type, supplier, internal number
filter: type ~ *erver*
csv: 0
----


===== Workstations =====
---- struct table ----
schema: system
cols: %title%, type, supplier, internal number
filter   : type ~ *orkstation*
csv: 0
----
```

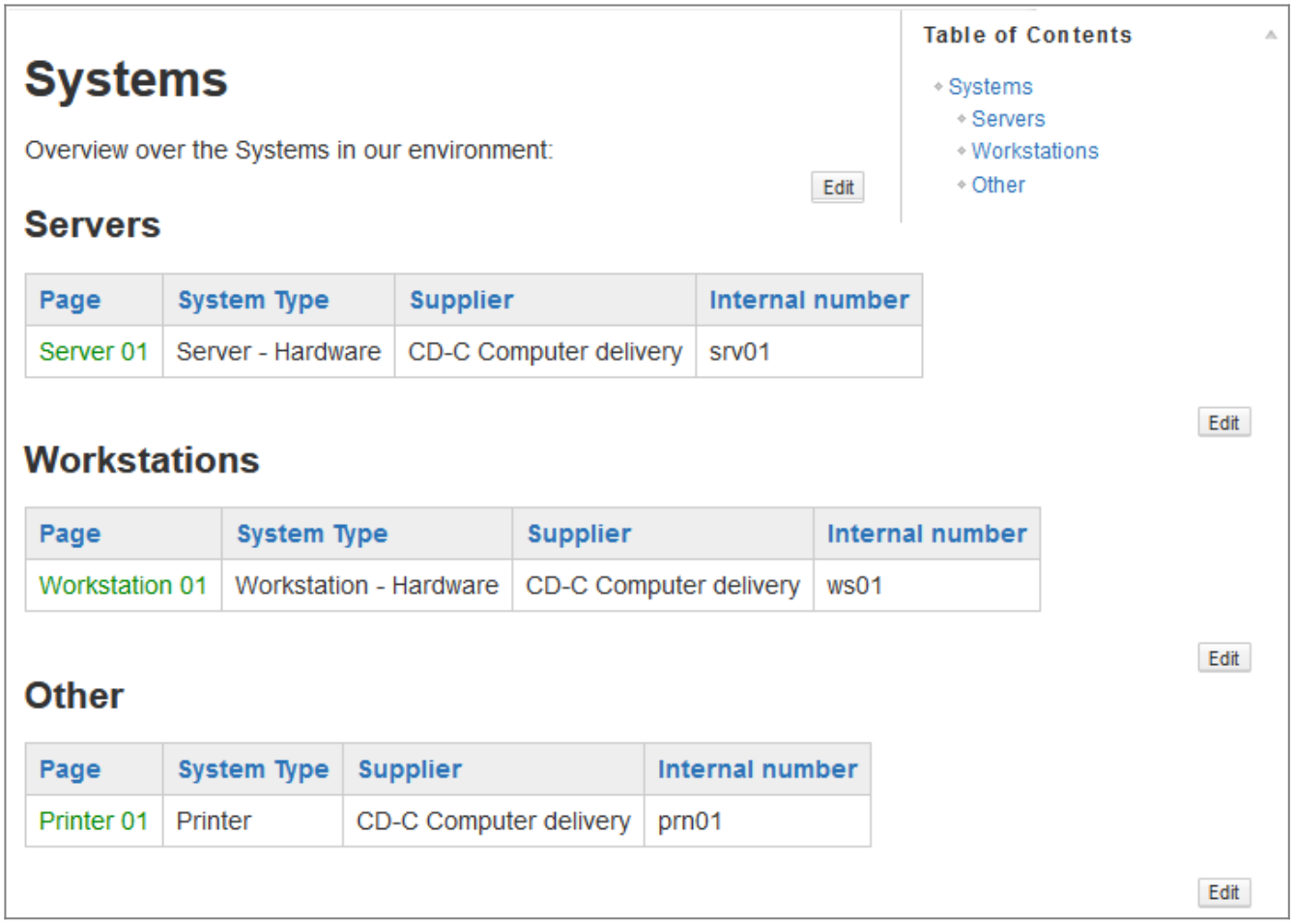
```
===== Other =====
---- struct table ----
schema: system
cols: %title%, type, supplier, internal number
filter: type !~ *erver*
filter: type !~ *orkstation*
csv: 0
----
```

This creates an overview of all entries in the schema `system`, categorized by server, workstation and others. This only works if the system types are named as in the example! Otherwise the filters have to be changed. We will come to that in a moment.

We already know Struct table, but now we use wildcard filters.

- `cols: %title%` we use a Special Column, Title is a Link to the page with the found data
- `filter: type ~ *erver*` We use the wildcard filters, here it doesn't matter if server is written with a capital or lowercase s. It also doesn't matter if there is something after server.
- `filter: type !~ *erver*` the same as before, only that it finds everything that is not a (S|s)erver
- `filter: type !~ *orkstation*` the second filter is 'anded', full reading of Other: Find anything that is not a server nor a workstation.

We get now a nice dynamic Overview:



New entries are automatically inserted in the corresponding category. So the overview is always up to date.

# Maintenance Overview (Dynamic Filters, Sum Calculation)

In order to create a more detailed overview of the maintenance, we extend the Maintenance schema by two fields. On the one hand, we want to go directly to the corresponding system and on the other hand, we want to record costs.

So we create the following additional fields:

Fields (**Configuration - only not default is mentioned**): Select first the Type of the field. Click «Save» after each Field, you get a new sort-number and a new row.

| **Schema: `maintenance`** | | | | | |
|---|---|---|---|---|---|
| **Sort** | **Field Name** | **Multi-Input?** | **Configuration** | **Type** | **Enabled** |
| 70 | systempage | false | label: System Page, usetitles: true, [**optional:** namespace: your system namespace] | Page | true |
| 80 | cost | false | label: Cost, roundto: 2 | Decimal | true |

After that, it is appropriate to adjust the Bureaucracy form as well:

```
===== Add Maintenance =====

<form>
action struct_lookup
struct_field "maintenance.date" =%Y-%m-%d
struct_field "maintenance.type"
struct_field "maintenance.description"
struct_field "maintenance.comment" !
struct_field "maintenance.cost"
struct_field "maintenance.user"
struct_fieldhidden "maintenance.system" "=[""@FORMPAGE_ID@"",0]"
struct_fieldhidden "maintenance.systempage" "=[""@FORMPAGE_ID@""]"
submit "Add Enty"
thanks "Please reload the Page!"
</form>
```

We get the new Cost field and fill the System Page field with the current page.

Optionally, we can also display the field: Cost in the «Maintenance» table of the system page:

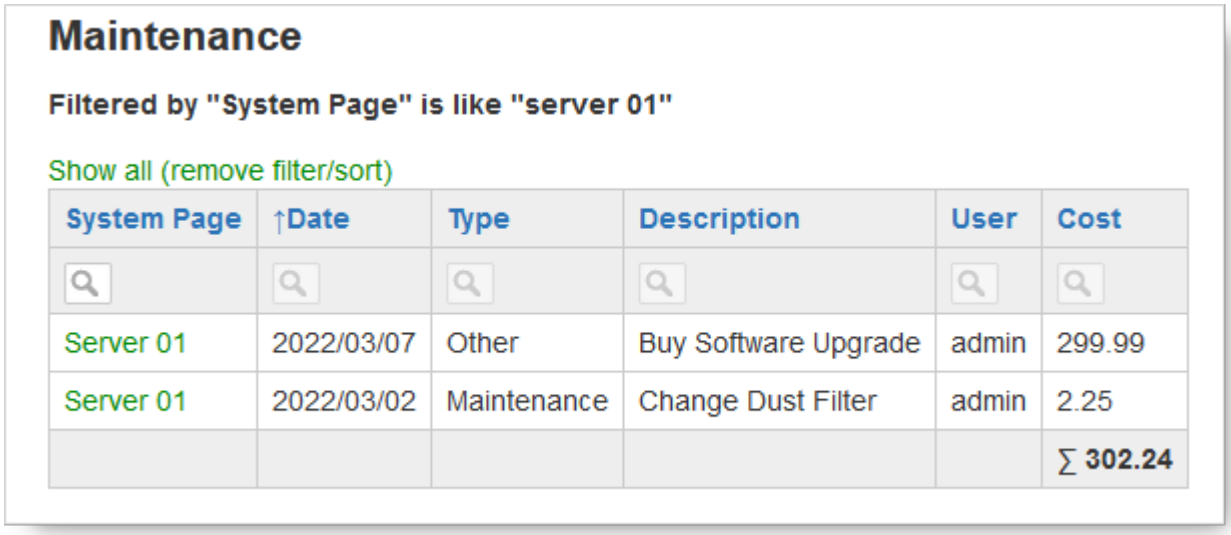
```
---- struct table ----
cols: date,type,description,comment,cost,user
schema: maintenance
filter   : system = $STRUCT.system.internal number$
sort: ^date
csv: 0
----
```

Now, perhaps on the overview page, we create a table that contains all the maintenance:

```
===== Maintenance =====

---- struct table ----
schema: maintenance
cols: systempage,date, type, description, user,cost
sort: ^date
dynfilters: 1
summarize: 1
csv: 0
----
```

We use dynamic filters and summation here.



**Maintenance**

**Filtered by "System Page" is like "server 01"**

Show all (remove filter/sort)

| System Page | ↑Date | Type | Description | User | Cost |
|---|---|---|---|---|---|
| Server 01 | 2022/03/07 | Other | Buy Software Upgrade | admin | 299.99 |
| Server 01 | 2022/03/02 | Maintenance | Change Dust Filter | admin | 2.25 |
| | | | | | ∑ 302.24 |

## Serial Data

As an alternative to maintenance in Medium Complex Sample, it is also possible to use serial data. These are always bound to the corresponding page. The procedure is similar, but we do not need a lookup field.

We can use the schema `maintenance` from the example or use a new optimized schema for it. For serial data, the order is determined by the order of the fields. We can either determine this via the sort property or create it correctly right away.

Fields (**Configuration - only not default is mentioned**): Select first the Type of the field. Click «Save» after each Field, you get a new sort-number and a new row.

| **Schema: `maintenance`** | | | | | |
|---|---|---|---|---|---|
| **Sort** | **Field Name** | **Multi-Input?** | **Configuration** | **Type** | **Enabled** |
| 10 | date | false | label: Date | Date | true |
| 20 | type | false | label: Type, values: «Maintenance, Repair, Other» | Dropdown | true |
| 30 | description | false | label: Description | LongText | true |

| Schema: maintenance | | | | | |
|---|---|---|---|---|---|
| **Sort** | **Field Name** | **Multi-Input?** | **Configuration** | **Type** | **Enabled** |
| 40 | comment | false | label: Comment | LongText | true |
| 50 | user | false | label: User, existingonly: false | User | true |

Instead of the system example we used, we only need the following code in the system page:

srv01.txt

```
====== Server 01 ======

===== Supplier Contact: =====
---- struct list ----
schema: systemsupplier
cols: main contact, email, phone
header: "Name: ", "- Mail: "," - Phone: "
sepbyheaders: yes
filter: name = $STRUCT.system.supplier$
----

===== Maintenance =====

---- struct serial ----
schema: maintenance
----
```

We also get a table, similar to the global data, with an input field. Easier to create and the data is automatically bound to the page. Only mandatory fields can not be defined this way.

From:
https://wwoss.ru/ - **worldwide open-source software**

Permanent link:
**https://wwoss.ru/doku.php?id=wiki:plugin:struct:sample:complexagg**

Last update: **2026/01/18 15:46**