TYPO3 in the larger world

Interacting with third-party applications

François Suter

TYPO3 Developer Days ‘08, Hamburg, 10th May 2008
Back in 2006

Table definition

Data sets

Specific search

Indexed search

Enumerated Lists

Filters

Menu integration

Views

Generic template

Joined records

Cross-linking

WFQBE by Mauro Lorenzutti

FE input

BE export

FE export
Change of focus

Data sets

Contextualisation

Views
- Lists
- Trees

Generic template
- Menu integration

Cross-linking

Joined records
From the inside: external import

Extension key: external_import

Features:

- *Define external data source using extended TCA syntax*
- *Backend end module for manual sync*
- *Gabriel integration for automated sync*
**From the inside: generic connectors**

New service type for standardising communication with third-party applications

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>init()</td>
<td>Validate connection</td>
</tr>
<tr>
<td>query()</td>
<td>Get data from distant source</td>
</tr>
<tr>
<td>fetchRaw()</td>
<td>Return data as is</td>
</tr>
<tr>
<td>fetchXML()</td>
<td>Return data as XML structure</td>
</tr>
</tbody>
</table>
External import: extended TCA

Demo
From the outside: remote server

Extension key: remote_server

Features:

- Receive calls from external applications in a standardised way
- Return responses in standard formats (XML, JSON)
- Based on BE Ajax script of TYPO3 4.2
- Secured access using existing TYPO3 auth services
Remote server

Demo
Gathering data: a rough tool

Extension: dataquery

Features:

- *Type a simple SQL statement*
- *Handling of enable fields is automated*
- *Perform outer joins*
- *Get standardised result set back*
- *Implement special features in query*
Filtering data: external tool

1st step

TYPO3 \(\rightarrow\) FOCUS

reads data from

FOCUS \(\rightarrow\) other Data sources

user tags the data

2nd step

TYPO3 \(\leftrightarrow\) FOCUS

sends context

TYPO3 \(\leftrightarrow\) FOCUS

returns identifiers
Standard dataset structure

array(
    'name' => 'maintable',
    'records' => array(
        ...
        'subtables' => array(
            'name' => 'subtable',
            'records' => array(
                ...
            )
        )
    )
)
Display engine

Extension datadisplay:

- New content element type
- Reads the standardised dataset structure
- Uses TypoScript for maximum flexibility
- Can handle joined records
- More of a proof of concept for now, will be expanded in the future
Display engine: example TS

```typescript
plugin.tx_datadisplay_pil {
    configs.foobar {
        allWrap.wrap = <table cellpadding="0" cellspacing="0" border="0">|</table>
        row.cObject = COA
        row.cObject {
            10 = TEXT
            10.value = my_field_1
            10.wrap = <td>|</td>
            20 = TEXT
            20.value = my_field_2
            20.wrap = <td>|</td>
            wrap = <tr>|</tr>
            30 < plugin.tx_datadisplay_pil
            30.userFunc = tx_datadisplay_pil->sub
            30 {
                name = subtable
                configs. >
                configs.subtable {
                    ...
                }
            }
            field >
        }
    }
}```
Display engine

Demo
Sending data back

Nothing definite yet, but:

- *a TCEmain hook could detect extended TCA syntax and write to external source*
- *issues of data integrity between two or more applications must be seriously considered*
- *data-routing software?*