pt_extbase

Content

General Information

pt_extbase provides different extensions and utilities for extbase developers. It also includes a set of viewHelpers for use in many purposes.

Administration

Integration

Development

ViewHelpers

Backend

Content

RenderPage

Renders the complete content of a given pageUid.

Arguments:

`pageUid`: The page Uid

Form

FormToken

Just returns a formToken to be used in backend form links

Arguments:

Example:

```
{ptx:be.formToken()}
```

Returns:

&formToken=<formTokenHash>

Format

RemoveLineBreaks

Remove line breaks from the string.

Arguments:

`string`: The string to remove the linebreaks from.

Example

```
{ptx:format.removeLineBreaks(string:'bla')}
```

FileSize

Formats a given integer as a fileSize.

Arguments:

`labels`: Labels for each degree in the notation "B| KB| MB| GB"

Example

<ptx:format.fileSize labels="B| KB| MB| GB">122132</ptx:format.fileSize>

StringToLower

Formats a string to lower characters.

Arguments:

`string`: Input string

TimeStamp

Formats an integer timestamp as date

Arguments:

`timestamp:` The timestamp integer
`format:` The format string for strftrime()

Javascript

Rbac

HasAccess

Checks the access right for a given role and object action.

Arguments:

`object:` Object to check if user has access rights for
`action:` Action to check if user has access rights for
`hasAny:` Check if user has access rights for any of these object:action combinations
`hasAll:` Check if user has access rights for all of these object:action combinations

Examples:

```
<code title="Basic usage">
        <rbac:hasAccess object="rbac_object_name" action="rbac_action_name">
                This is being shown in case user has access to action on object
        </rbac:hasAccess>
</code>
```

Everything inside the <rbac:access> tag is being displayed if the frontend user has access to action on object. If no user is given, the currently logged in fe user will be used.

```
<code title="hasAccess / access / noAccess">
        <rbac:hasAccess object="rbac_object_name" action="rbac_action_name">
                <f:then>
                        This is being shown in case the user has access.
                </f:then>
                <f:else>
                        This is being displayed in case the user has NO access.
                </f:else>
        </rbac:hasAccess>
</code>
```

Everything inside the "access" tag is displayed if the user has access to action on object. Otherwise, everything inside the "noAccess"-tag is displayed.

```
<code title="inline notation">
        {rbac:hasAccess(object: 'objectName', action: 'actionName'
                then: 'user has access', else: 'access is denied')}
</code>
```

The value of the "then" attribute is displayed if access is granted for user on object and action. Otherwise, the value of the "else"-attribute is displayed.

Tree

Path

Iterates over the path from a given node Id to the root, to draw a path or a rootline menu.

Adds the following variables to the template:

node: current node
firstNode: Boolean, true if first node

Arguments:

repository: Specifies the node repository
namespace: Specifies the tree namespace
node: The node uid
skipRoot: Skip the root node

Example:

```
<f:for each="{ptx:explode(delimiter: ',', string:listRow.categoryUid.value.categoryUid)}" as="categoryUid">
  <div>
   <ptx:tree.path node="{categoryUid}" skipRoot="1" namespace="tx_ptcertification_domain_model_category"
                repository="Tx_PtCertification_Domain_Repository_CategoryRepository" >
   <f:if condition="{firstNode}">
         <f:then>{node.label}</f:then>
         <f:else>&raquo; {node.label}</f:else>
   </f:if>
  </ptx:tree.path>
  </div>
</f:for>
```

Selector

Renders a javascript tree selector

Arguments:

repository: Specifies the node repository
namespace: Specifies the tree namespace
node: The node uid
skipRoot: Skip the root node

- repository Repository class name to be used as node repository (not as tree repository!)
- namespace Namespace for which to create tree
- name Name of the form field (see input.text viewhelper!)
- value Uid of selected node (if in 1:N mode) or comma separated list of UIDs (if in M:N mode)
- multiple If set to 1, multiple nodes can be selected in widget
- restrictedDepth If a value is given, tree is only rendered to given depth (1 = only root node is rendered)

Example:

```
<f:for each="{ptx:explode(delimiter: ',', string:listRow.categoryUid.value.categoryUid)}" as="categoryUid">
  <div>
   <ptx:tree.path node="{categoryUid}" skipRoot="1" namespace="tx_ptcertification_domain_model_category"
                repository="Tx_PtCertification_Domain_Repository_CategoryRepository" >
```

```
   <f:if condition="{firstNode}">
         <f:then>{node.label}</f:then>
         <f:else>&raquo; {node.label}</f:else>
   </f:if>
  </ptx:tree.path>
  </div>
</f:for>
```

Misc

Comment

Just removes everything between the tags.

Example:

```
<ptx:comment>
<!--
 Here comes the comment which is never rendered.
-->
</ptx:comment>
```

Explode

Explodes a string by the given delimiter.

Arguments:

`delimiter`: The delimiter character
`string`: The string to explode.

Example

```
{ptx:explode(delimiter: ',', '1,2,3,4')}
```

Captcha

Uses the *captcha* Extension to render a captcha image.

Example

```
<ptx:captcha />
```

CObjectConfig

Renders the given cObject through cObjGetSingle.

Arguments:

`config`: The cObject config
`data`: Optional data to use for rendering the cObejct.

Example

```
{ptx:cObjectConfig(config: config)}
```

IfValueChanges

ViewHelper evaluates to true every time a new value is given. This can be used in a loop for example to render a header when one dimension of the objectcollection changes.

Example:

```
<f:for each="{things}" as="thing">
        <ptx:ifValueChanges value="{thing.category}">
                <h1>{thing.category}</h1>
        </ptx:ifValueChanges>
</f:for>
```

RequestArguments

Shows a submitted extension argument.

Arguments:

key: The argument array key

Example:

```
<ptx:requestArguments key="action" />
```

Assertions

The static class `Tx_PtExtbase_Assertions_Assert` which includes different assertion functions.

Example: Test if a value is a positive integer. The second parameter denotes if the integer is allowed to be false:

```
Tx_PtExtbase_Assertions_Assert::isPositiveInteger($var, FALSE, array('message' => 'The given value has to be a positive integer');
```

Configuration

Abstract Configuration Builder

The configurationBuilder can be implemented in your extension to:

- Evaluate your typoscript settings at ONE single point before running your actual extension-code
- Build configuration objects with working code code completion within the IDE
- Access this everywhere in the code

The configuration builder works as registry for the configuration objects. It builds these objects on its first access and the serves them on further requests from its local cache.

Abstract Configuration

The abstract configuration can be used for implementing your extension configuration objects. The derived object has to implement the `init()` function, which is then called from the configurationBuilder during the init process. The init function evaluates and sets the properties of this object.

The following example shows an implementation of the AbstractConfiguration. The variable `useSession` is define ind the header and set to a default value. Within the init section this is value is set to a true boolean value with the utility function `setBooleanIfExistsAndNotNothing` which only sets the variable if it is set in the settings.:

```
    class Tx_PtExtlist_Domain_Configuration_Base_BaseConfig
            extends Tx_PtExtbase_Configuration_AbstractConfiguration {


    /**
     * @var bool
```

```
     */
    protected $useSession = FALSE;



    /**
     * Template method for initializing this config object by injected
     * TypoScript settings.
     *
     * @return void
     */
    protected function init() {
            $this->setBooleanIfExistsAndNotNothing('useSession');
    }


    /**
     * @return bool
     */
    public function getUseSession() {
            return $this->useSession;
    }

}
```

These utility functions are available:

- `setValueIfExists($tsKey, $internalPropertyName = NULL)`: The first value is the key from the typoscript array, the second optional defines the internal property name if it differs from the settings key.
- `setValueIfExistsAndNotNothing($tsKey, $internalPropertyName = NULL)`
- `setBooleanIfExistsAndNotNothing($tsKey, $internalPropertyName = NULL)`
- `setRequiredValue($tsKey, $errorMessageIfNotExists, $internalPropertyName = NULL)`: If this method is used and the value is not present, an exception is thrown. The second parameter defines the exception message.

Controller

AbstractActionController

The abstraction action controller extends the extbase ActionController and adds some new functions and behaviours.

Exchange Single Fluid Templates and Views

Single fluid templates can be exchanged by TypoScript configuration by using the following syntax:

```
plugin.<plugin_key>.settings.controller.<Controller_Name_Without_Controller>.<action_name_without_action>.template = full_path_to_template_with.html
```

View can be set via TS. View has to be set in TS via:

```
plugin.<plugin_key>.settings.controller.<Controller_Name_without_Controller>.<action_Name_without_Action>.view = ViewClassName
```

Lifecycle manager

The lifecycle manager can be used to trigger actions during the extensions lifecycle. This for example is used to automatically restore the session array at the beginning of the extensions lifecycle and to store it to the database again at the end.

Utility

FakeFrontendFactory

FakeFrontendFactory enables the use of frontend functionality like cObj-Rendering in the backend. To fake a frontend you just have to call the factory:

t3lib_div::makeInstance('Tx_PtExtbase_Utility_FakeFrontendFactory')->createFakeFrontend();

HeaderInclusion

NameSpace

Tca

AjaxDispatcher

eIDDispatcher

4. Schnittstellen

Die Extension bindet keine Schnittstellen ein und stellt keine Schnittstellen zur Verfügung.

Changelog

V 1.2.0:

- ADD: Compatibility layer for TYPO3 6.0

V 1.0.0:

- ADD: Tree, a framework to handle nested-set-trees.
- CHG: Some changes to the abstract configuration builder.

V 0.0.3:

- FIX: Fixed a bug in namespace utility, which was responsible for unused array keys in the session array. (Broke realurl config)