

TYP03 Extbase Cheat Sheet 1

Git:
Forge:
Extension Builder:

git clone <http://git.typo3.org/TYP03v4/CoreProjects/MVC/extbase.git>
<http://forge.typo3.org/projects/show/typo3v4-mvc>
http://git.typo3.org/TYP03v4/Extensions/extension_builder.git



v 2.05 / 21.02.2012
Patrick Lobacher
www.typovision.de

typovision://

Domain Driven Design (terms by wikipedia)

Domain	A sphere of knowledge, influence, or activity. The subject area to which the user applies a program is the domain of the software.
Model	A system of abstractions that describes selected aspects of a domain and can be used to solve problems related to that domain.
Ubiquitous Language (UL)	A language structured around the domain model and used by all team members to connect all the activities of the team with the software. The words of the UL are used throughout all artefacts.
Context	The setting in which a word or statement appears that determines its meaning.
Entity	An object that is not defined by its attributes, but rather by a thread of continuity and its identity. Derives from class Tx_Extbase_DomainObject_AbstractEntity
Value Object (VO)	An object that contains attributes but has no conceptual identity. They should be treated as immutable. Derives from class Tx_Extbase_DomainObject_AbstractValueObject
Aggregate	A collection of objects that are bound together by a root entity, otherwise known as an aggregate root.
Aggregate root	The aggregate root guarantees the consistency of changes being made within the aggregate by forbidding external objects from holding references to its members.
Service	When an operation does not conceptually belong to any object. Following the natural contours of the problem, you can implement these operations in services.
Repository	Methods for retrieving domain objects should delegate to a specialized Repository object such that alternative storage implementations may be easily interchanged. Derives from class Tx_Extbase_Persistence_Repository

Naming Conventions

Classes	UpperCamelCase
Methods & Variables	lowerCamelCase
General class naming	Tx_{ExtensionName}_{Path}.php - {ExtensionName} is extension_key without _ and in UpperCamelCase - {Path} is inside Classes directory of the extension, change / with _ e.g.: Class-Name: Tx_BlogExample_Domain_Model_Post Path & Filename: typo3conf/ext/blog_example/Classes/Domain/Model/Post.php or Class-Name: Tx_Extbase_MVC_Controller_ActionController Path & Filename: typo3sys/ext/extbase/Classes/MVC/Controller/ActionController.php
Interfaces	Class name ends in „Interface“, e.g. Tx_Extbase_MVC_RequestInterface
Abstract classes	beginning of last class name part is „Abstract“ e.g. Tx_Extbase_MVC_Controller_AbstractController Domain model entity: ... extends Tx_Extbase_DomainObject_AbstractEntity Domain model value object: ... extends Tx_Extbase_DomainObject_AbstractValueObject

PHPDoc annotations

@param [Type] \$var	Action: Parameter. Svar validates to [Type]
@dontvalidate \$var	Action: No validation for Svar (needed for new und edit actions)
@var [Type]	Model: Type of var in Domain Model - either simple type, class or ObjectStorage: Tx_Extbase_Persistence_ObjectStorage<Tx_{ExtensionName}_Domain_Model_{Model}> delivers methods: count(), attach(), attachAll(), detach(), detachAll(), contains(), ...
@validate [\$var] [Validator]	Model & Action: Validation for Svar. In model without Svar.
@lazy	Lazy loading in domain model (load child objects only when needed)
@cascade remove	Delete child(s) if parent is removed
@dontverifyrequesthash	Disable request hash checking
@return [Type]	Return value is of type [Type]
@api	Declares that the following class/method is part of the official API
@deprecated	Declares that the following class/method should not be used anymore
@scope	Defines the type of the class. Possible values are prototype and singleton Not used in the moment: For singleton use ...implements t3lib_Singleton instead

Folder structure inside extension dir typo3conf/ext/{extension_key}/

ext_autoload.php	Mapping classname to classfile location
ext_emconf.php	Extension manager configuration
ext_icon.gif	Extension icon
ext_localconf.php	Frontend configuration
ext_tables.php	Backend configuration
ext_tables.sql	SQL statements for the database structure
ExtensionBuilder.json	ExtensionBuilder configuration
Classes/	All PHP classes reside here
Classes/Controller/	Controller classes
Classes/Controller/{DomainObjectName}Controller.php	Controller of model {DomainObjectName} (rec.)
Classes/Domain/	Domain specific classes
Classes/Domain/Model/	Model classes
Classes/Domain/Model/{DomainObjectName}.php	Specific model class for {DomainObjectName}
Classes/Domain/Repository/	Repository classes
Classes/Domain/Repository/{DomainObjectName}Repository.php	Repository of model {DomainObjectName}
Classes/Domain/Validator/	Validator classes
Classes/Domain/Validator/{DomainObjectName}Validator.php	Validator of model {DomainObjectName}
Service	Service classes
Utility	Utility classes (just helper classes)
Classes/ViewHelpers/	Individual fluid view helpers reside here
Classes/ViewHelpers/{VHName}ViewHelper.php	View helper with name VHName
Configuration/	All configuration (structure is a suggestion)
Configuration/TCA/	Table configuration array (TCA). Files e.g.: {DomainObjectName}.php
Configuration/FlexForms/	FlexForms used for backend forms
Configuration/TypoScript/	TypoScript constants and setup (e.g. constants.txt and setup.txt)
Documentation/	All documentation reside here
Documentation/Manual/	Extension manual, subfolder {format}/{lang}/
Resources/	All resources reside here
Resources/Private/	Private resources
Resources/Private/Backend/	Resources used by backend modules
Resources/Private/Backend/Layouts/	Layout files for backend modules
Resources/Private/Backend/Templates/	Template files for backend modules
Resources/Private/Backend/Templates/{ControllerName}/	All templates of a specific controller (BE)
Resources/Private/Backend/Templates/{ControllerName}/[action].format	Template of [action] from [Controller] (BE)
Resources/Private/Language/	Language files for l10n
Resources/Private/Language/locallang.xml	Main language file - use key w. translate viewhelper
Resources/Private/Layouts/	Layout files for frontend plugins
Resources/Private/Partials/	Partials files for frontend plugins
Resources/Private/Templates/	Template files for frontend plugins
Resources/Private/Templates/{Controller}/	All templates of a specific controller (FE)
Resources/Private/Templates/{Controller}/[Action].format	Template of [Action] from [Controller] (FE)
Resources/Public/	Additional resources (own dirs if needed, like „Icons“, ...)
Tests/	All tests reside here
Tests/Unit/	Unit tests

Extension-API

Frontend Plugins	
ext_localconf.php	Tx_Extbase_Utility_Extension::configurePlugin(\$EXTKEY, 'pi1', // plugin name array('Controller1' => 'action1, action2, ...', // controller action 'Controller2' => 'action3, action4, ...', // combinations ...), array('Controller1' => 'action1, action6, ...', // controller action 'Controller2' => 'action5, action4, ...', // combinations uncached ...), Tx_Extbase_Utility_Extension::TYPE_PLUGIN // or TYPE_CONTENT_ELEMENT);
ext_tables.php	Tx_Extbase_Utility_Extension::registerPlugin(\$EXTKEY, 'pi1', // plugin name 'Plugin title', // plugin title t3lib_extMgm::extRelPath(\$EXTKEY) . 'Resources/Public/Icons/someIcon.gif');
Backend Module	
ext_tables.php	Tx_Extbase_Utility_Extension::registerModule(\$EXTKEY, 'web', // main module 'tx_extkey_m1', // sub module \$position, // see below array('Controller1' => 'action1, action2, ...', // controller action 'Controller2' => 'action3, action4, ...', // combinations ...), // configuration array array('access' => 'user,group', 'icon' => 'EXT:extkey/ext_icon.gif', 'labels' => 'LLL:EXT:extkey/Resources/Private/Language/...')); Position has this syntax: [cmd]:[submodule-key]. cmd can be „after“, „before“ or „top“ (or blank - default). If „after“/„before“ then submod will be inserted after/before the existing submod with [submodule-key] if found. If not, the bottom of list. If „top“ the module is inserted in the top of the submodule list.
General	
ext_tables.php	// Static TypoScript t3lib_extMgm::addStaticFile(\$EXTKEY, 'Configuration/TypoScript', 'TemplateName'); // Include flexforms \$TCA['tt_content']['types']['list']['subtypes_addlist'][\$EXTKEY . '_pi1'] = 'pi_flexform'; t3lib_extMgm::addPiFlexFormValue(\$EXTKEY . '_pi1', 'FILE:EXT: . \$EXTKEY . 'Configuration/FlexForms/flexform.xml'); // Allow data entries on standard pages t3lib_extMgm::allowTableOnStandardPages(['tablename']); // TCA - see separate section

MISC

FlashMessages (\$this->flashMessageContainer->)	
add(\$message,\$title,\$severity)	Add another flash message. t3lib_FlashMessage::NOTICE, t3lib_FlashMessage::INFO, t3lib_FlashMessage::OK, t3lib_FlashMessage::WARNING, t3lib_FlashMessage::ERROR
getAllMessages()	Get all flash messages currently available
flush()	Reset all flash messages
getAllMessagesAndFlush()	Get all flash messages currently available. And removes them from the session
Exceptions	
throw new Exception('ErrorMessage' , UNIX-TIMESTAMP);	// Unix-TS because of uniqueness
Persistence Manager (persist all objects) - done with DI	
function injectPersistenceManager(Tx_Extbase_Persistence_Manager \$persistenceManager) { ... \$persistenceManager->persistAll(); }	
Object Manager (DI)	
function injectObjectManager(Tx_Extbase_Object_ObjectManagerInterface \$objectManager) { ... \$objectManager->get('Tx_ExtKey_Classname'); \$objectManager->create('Tx_ExtKey_Classname'); }	
Translation	
Tx_Extbase_Utility_Localization::translate(\$label, '{ExtensionName}');	



TypoScript		
CONSTANTS: plugin.tx_[lowercasedextensionname] & module.tx_[lowercasedextensionname]		
view	# cat=plugin.tx_myextension/file; type=string; label=Path to template root (FE) templateRootPath = EXT:my_extension/Resources/Private/Templates/	
persistence	# cat=plugin.tx_myextension/;a; type=int+; label=Default storage PID storagePid =	
SETUP: plugin.tx_[lowercasedextensionname] & module.tx_[lowercasedextensionname] & config.tx_extbase		
settings		Access in controller: \$this->settings Access in Fluid: (settings)
persistence	enableAutomaticCacheClearing storagePid	Cache clearing at edit/insert operations PID list where records are read from
persistence. classes. [fullClassName].	newRecordStoragePid mapping.tableName mapping.recordType	PID where new records will be stored Which table is mapped Record type (needs TCA field 'type' => 'record_type'),
	mapping.columns.	field_name mapOnProperty = propertyName
	subclasses.	put one entry for every subclass in superclass definition [Identifier] = [fullClassName]
view	templateRootPath partialRootPath layoutRootPath	Template path Parial path Layout path
_LOCAL_LANG	[ISOlang].key [default].key	Localization (key corresponds to the key in file: Resources/Private/Language/locallang.xml)
_CSS_DEFAULT_STYLE		Inline stylesheets
SETUP: config.tx_extbase (Dependency Injection DI)		
objects. [sourceClass]. className = [targetClass]	At all places where the code refers to [sourceClass], an object of [targetClass] should be instantiated.	lib.foo = USER lib.foo { userFunc = tx_extbase_core_bootstrap->run extensionName = YourExtension pluginName = YourPlugin switchableControllerActions { Standard { 1 = action2

ActionController API (Tx_Extbase_MVC_Controller_ActionController)	
\$this->actionMethodName	Name of current action
\$this->argumentsMappingResults	Results of the argument mapping (will be used in errorAction)
\$this->defaultViewObjectName	Name of default view, if no other view is found
\$this->errorMethodName	Name of error action - default is „errorAction“
\$this->request	Request object (of type Tx_Extbase_MVC_RequestInterface)
\$this->response	Response object (of type Tx_Extbase_MVC_ResponseInterface)
\$this->settings	Domain specific settings from TypoScript (array)
\$this->view	current view (of type Tx_Extbase_MVC_View_ViewInterface)
\$this->viewObjectNamePattern	Tx_@extension_view_@controller_@action_@format
function [actionName]Action()	Specific method for action [actionName]
function errorAction()	Called at arguments validation error.
\$this->forward(\$actionName, \$controllerName, \$extensionName, array \$arguments)	Internal redirect of request to another controller
function initializeAction()	Initialize method for all actions
function initializeActionNameAction()	Initialize method for specific action [ActionName]
function initializeView(Tx_Extbase_MVC_View_ViewInterface \$view)	Initialize method for the committed view
\$this->redirect(\$actionName, \$controllerName, \$extensionName, array \$arguments, \$pageUid, \$delay = 0, \$statusCode = 303)	External HTTP redirect to another controller
\$this->redirectToUri(\$uri, \$delay=0, \$statusCode=303)	Redirect to URI
function resolveView()	Can be used for creating an own view object
function resolveViewObjectName()	Resolves name of view object if no fluid template is found
\$this->throwStatus(\$statusCode, \$statusMessage, \$content)	Send HTTP status code

View API (Tx_Extbase_MVC_View_ViewInterface)	
\$this->view->assign(\$key,\$value)	Assign \$value to key \$key in view
\$this->view->assignMultiple(array \$values)	Assign array \$values view - use key of array element for key in view
\$this->view->initializeView()	Initializing the concrete view implementation
\$this->view->render()	Returns the rendered view

Request API (Tx_Extbase_MVC_RequestInterface) \$this->request		
get set	ControllerExtensionName(\$extensionName)	Gets/Sets the extension name of the controller
get set	ControllerName(\$controllerName)	Gets/Sets the name of the controller
get set	ControllerActionName(\$actionName)	Gets/Sets the name of the action
get set	Argument(\$argumentName, \$value)	Gets/Sets the value of the specified argument
get set	Arguments(array \$arguments)	Gets/Sets the whole arguments array
	hasArgument(\$argumentName)	Checks if an argument of the given name exists (is set)
get set	Format(\$format)	Gets/Sets requested representation format
get set	Errors(array \$errors)	Gets/Sets the request errors

Repository API (Tx_Extbase_Persistence_Repository)	
<pre> \$[domainModelName]Repository = t3lib_div::makeInstance("Tx_Extbase_Domain_Repository_[DomainModelName]Repository"); </pre>	
<pre> * @param Tx_ExtName_Domain_Repository [DomainModelName]Repository \$[domainModelName]Repository public function inject(DomainModelName)Repository(Tx_ExtName_Domain_Repository [DomainModelName]Repository \$[domainModelName]Repository) { \$this->[domainModelName]Repository = \$[domainModelName]Repository; } </pre>	
add(\$object)	Add object to repository
countAll()	Returns the total number objects of this repository
countBy[PropertyName](\$propertyValue)	Returns the number objects with [PropertyName] == \$propertyValue
findByUid(\$uid)	Finds an object matching the given identifier
findAll()	Find all objects of given type
findBy[PropertyName](\$propertyValue)	Find all objects where property [PropertyName] == \$propertyValue
findOneBy[PropertyName](\$propertyValue)	Same as above, just find one (the first found) object (type object!)
remove(\$object)	Remove object from repository
removeAll()	Removes all objects of this repository
replace(\$existingObject, \$newObject)	Replaces an existing object with a new one
update(\$object)	Update stored object with Subject

Custom queries (use in own method inside repository)	
<pre> \$query = \$this->createQuery(); return \$query->matching(\$query->logicalAnd(\$query->equals('blog', \$blog), \$query->equals('tags.name', \$tag)))->setOrderings(array('date' => Tx_Extbase_Persistence_QueryInterface::ORDER_DESCENDING))->setLimit((integer)\$limit)->execute(); // \$defaultPersistences </pre>	
\$query = \$this->createQuery();	Initializes a query. Access to below methods with \$query->...
logicalAnd(\$constraints)	Performs a logical conjunction of the given constraints
logicalOr(\$constraints)	Performs a logical disjunction of the given constraints
logicalNot(\$constraint)	Performs a logical negation of the given constraint
statement(\$constraint)	SQL-Statement
setOrderings(array \$orderings)	Tx_Extbase_Persistence_QueryInterface::ORDER_ASCENDING Tx_Extbase_Persistence_QueryInterface::ORDER_DESCENDING
setLimit(\$limit)	Sets the maximum size of the result set to limit (integer!)
setOffset(\$offset)	Sets the start offset of the result set to offset (integer!)
count()	Executes the query against the database and returns the number of matching objects
execute()	Executes the query against the backend and returns the result
getFirst()	Get the first result back
matching(\$constraints)	The constraint used to limit the result set. Use methods below...
equals(\$propertyName, \$operand, \$caseSensitive = TRUE)	Returns an equals criterion used for matching objects against a query
like(\$propertyName, \$operand)	Returns an equals criterion used for matching objects against a query
contains(\$propertyName, \$operand)	It matches if the multivalued property contains the given operand
in(\$propertyName, \$operand)	It matches if the property's value is contained in the multivalued operand
lessThan(\$propertyName, \$operand)	Returns a less than criterion used for matching objects against a query
lessThanOrEqual(\$propertyName, \$operand)	Returns a less or equal than criterion used for matching objects against a query
greaterThan(\$propertyName, \$operand)	Returns a greater than criterion used for matching objects against a query
greaterThanOrEqual(\$propertyName, \$operand)	Returns a greater than or equal criterion used for matching objects against a query

Flexform (example option maxPosts and switchableControllerActions) locallang.xml (in Resources/Private/Language)	
<pre> <T3DataStructure> <sheets> <sheet> <ROOT> <TEForms> <sheetTitle>LLL:EXT:my_extension/Resources/Private/Language/locallang_db.xml:ff.sheetTitle</sheetTitle> </TEForms> <type>array</type> </sheet> </ROOT> </sheets> </T3DataStructure> </pre>	<pre> <switchableControllerActions> <TEForms> <label>Some label</label> <config> <type>select</type> <items type="array"> <numIndex index="0">Default</numIndex> <numIndex index="1">Controller1->action1; Controller2->action2;...</numIndex> </items> <maxItems>1</maxItems> <size>1</size> </config> </TEForms> </switchableControllerActions> </pre>
<pre> ...ELEMENTS... </el> </ROOT> </sDEF> </sheets> </T3DataStructure> </pre>	<pre> <?xml version="1.0" encoding="utf-8" standalone="yes" ?> <T3locallang> <meta type="array"> <type>module</type> <description>Some description</description> </meta> <data type="array"> <languageKey index="default" type="array"> <label index="key1">Labels1</label> </languageKey> </data> </T3locallang> </pre>
<pre> <settings.maxPosts> <TEForms> <label>Max. number of posts</label> <config> <type>input</type> <size>2</size> <eval>int</eval> <default>10</default> </config> </TEForms> </settings.maxPosts> </pre>	<pre> <?xml version="1.0" encoding="utf-8" standalone="yes" ?> <T3locallang> <meta type="array"> <type>module</type> <description>Some description</description> </meta> <data type="array"> <languageKey index="default" type="array"> <label index="key1">Labels1</label> </languageKey> </data> </T3locallang> </pre>

Validator API (Tx_Extbase_Validation_Validator_ValidatorInterface)	
In annotation of Domain Model: Validator class for Domain Model:	@validate Validator1, Validator2(operand1 = value1, ...), ... class Tx_ExtName_Domain_Validator_[DomainModelName]Validator extends Tx_Extbase_Validation_Validator_AbstractValidator
Validation of controller arguments:	@validate \$variableName Validator1, Validator2, ...
Alphanumeric	TRUE, if the given property is a valid alphanumeric string, which is defined as [a-zA-Z0-9]*
DateTime	Checks if the given value is a valid DateTime object
EmailAddress	Checks if the given value is a valid email address
Float	Checks if the given value is a valid float
GenericObject	Checks if the given value is valid according to the property validators
Integer	Checks if the given value is a valid integer
NotEmpty	Checks if the given value is not empty (NULL or empty string)
NumberRange(startRange,endRange)	Returns TRUE, if the given value is a valid number in the given range
Number	Checks if the given value is a valid number
RegularExpression(regularExpression)	Returns TRUE, if the given value matches the given regular expression
StringLength(minimum,maximum)	Returns TRUE, if the given property (Svalue) is a valid string and its length
String	Returns TRUE, if the given property (Svalue) is a valid string
Text	Returns TRUE, if the given property (SpropertyValue) is a valid text (contains no XML tags)
Own Validators (use @validate Tx_ExtName_Domain_Validator_ValNameValidator(option1=value1,...))	
class Tx_ExtName_Domain_Validator_ValNameValidator extends Tx_Extbase_Validation_Validator_AbstractValidator { public function isValid(\$elem) { \$option1 = \$this->options['option1']; // options access \$this->addError('ErrorString', 1262341470); // type Tx_Extbase_Validation_Error return TRUE; // or FALSE // validates if TRUE } }	

UriBuilder API (Tx_Extbase_MVC_Web_Routing UriBuilder)	
Access in controller via \$this->uriBuilder	
<pre> section = "" format = "" createAbsoluteUri = FALSE addQueryString = FALSE linkAccessRestrictedPages = FALSE argumentsToBeExcludedFromQueryString = array() targetPageUid = NULL targetPageType = 0 noCache = FALSE useCacheHash = TRUE </pre>	UriBuilder Options there are Getter and Setter for all of them
setRequest(Tx_Extbase_MVC_Web_Request \$request) / getRequest()	Sets/Gets request
reset()	Resets all UriBuilder options to their default value
uriFor(\$actionName = NULL, \$controllerArguments = array(), \$controllerName = NULL, \$extensionName = NULL, \$pluginName = NULL)	Creates an URI used for linking to an Extbase action
build()	Builds the URI

TYPO3 Fluid Cheat Sheet 1

GIT: [git clone http://git.typo3.org/FLOW3/Packages/Fluid.git](http://git.typo3.org/FLOW3/Packages/Fluid.git)
 Forge: <http://forge.typo3.org/projects/show/package-fluid>
 Issue-Tracker: <http://forge.typo3.org/projects/typo3v4-mvc/issues>



v 2.05 / 21.02.2012
 Patrick Lobacher
www.typovision.de

typovision://

FLUIDTEMPLATE (TypeScript cObject)

10 = FLUIDTEMPLATE 10.file = fileadmin/templates/mytemplate.html 10.partialRootPath = fileadmin/templates/partials/ 10.variables { var1 = cObject ... }	Properties: file, extbase.pluginName, extbase.controllerName, extbase.controllerExtensionName, extbase.controllerActionName, layoutRootPath, partialRootPath, variables, format
--	---

Addressing the view in action (controller class)

\$this->view->assign('identifier', \$object)	Makes Subject (which is of kind string, array or object) available as {identifier} in fluid
\$this->view->render()	Forces rendering of template (default is at script end)

Templates, Layouts, Partials (in directory Resources/Private/...)

Template (Templates/[Controller]/[Action].html)	<pre><f:layout name="default" /> <f:section name="content"> <f:render partial="PartName" arguments="..." /> </f:section></pre>	Layout (Layouts/default.html)	<pre><f:render section="content" /></pre>
		Partial (Partials/PartName.html)	...

Object accessor syntax

{name.property}	Object accessor: Result of getProperty() in model {name}
{name.key}	Associative array: Element in array {name} with {key}
{name.number}	Numeric array: Element in array {name} at position {number}

ViewHelper syntax

{namespace}	Declares the abbreviation f as namespace for Tx_Fluid_ViewHelpers (this is default). You can add own namespaces.
<f:vhname PARAMS>	ViewHelper with the name {vhname}. Corresponding class is found at: typo3/sysext/fluid/Classes/ViewHelpers/VhnameViewHelper.php
PARAMS	Arguments will be listed like: {key1: value1, key2: value2} e.g. each="{0:1,1:2,...}" or values="{0:'odd',1:'even',...}" or arguments="{name:object}"

Inline syntax:
 <f:vhname ... /> could be written as {f:vhname(...)} and
 <f:format.n12br><f:format.crop maxCharacters="20">{some.value}</f:format.crop></f:format.n12br>
 could be written as {some.value -> f:format.crop(maxCharacters:20) -> f:format.n12brf}

Boolean expressions - like: XXX,YYY is of type number, object accessor, array or ViewHelper inline syntax
 condition="XXX operator YYY" operator is one of: == != % >= <= <

TagBasedViewHelper (general arguments for tag based viewhelper)

used with viewhelper: form (and all sub viewhelpers), image, link, renderFlashMessages

additionalAttributes	Associative array of additional tag-attributes	style	Individual CSS style
class	CSS classes for this tag	title	Tooltip text for this element
dir	Sets dir attribute - ltr or rtl	accesskey	Defines access key
id	Unique id	tabindex	Tab order for this element
lang	Language of this element (RFC 1766)	onclick	JavaScript for onclick event

alias ViewHelper - define alias of variables

<f:alias map="{x: foo.bar.baz, y: foo.bar.baz.name}">{x.name} or {y}</f:alias>	
map	Mapping of alias - array - just valid inside alias tags

base ViewHelper - returns a base href tag

cObject ViewHelper - display TypeScript object (access with data & current)

<f:cObject typscriptObjectPath="lib.someLibObject">{article}</f:cObject>	
<f:cObject typscriptObjectPath="lib.customHeader" data="{article}" currentValueKey="{article.title}" />	
typscriptObjectPath	TypeScript object path which should be displayed
data	Data which is used for the rendering - same as <f:cObject>data</f:cObject>
currentValueKey	Key which sets the stdWrap property current

count ViewHelper - counts elements

<f:count subject="{0:1, 1:2, 2:3, 3:4}" />	
subject	The array or ObjectStorage to iterated over

cycle ViewHelper - iterates through given values

<pre><f:for each="{0:1, 1:2, 2:3, 3:4}" as="foo"> <f:cycle values="{0:'odd', 1:'even'}" as="zebraClass"><li class="{zebraClass}">{foo}</f:cycle> </f:for></pre>	
values	Array of values which is used for iteration
as	Name of iteration variable

debug ViewHelper - wrapper for TYPO3 debug function

<f:debug>{testVariables.array}</f:debug>	
title	Title for debug output

escape ViewHelper - escapes special characters

<f:escape>{text}</f:escape>	<f:escape type="entities">{text}</f:escape>
value	Content which is escaped (for use with inline syntax)
encoding	Encoding (UTF-8 per default)
type	One of 'html', 'url' or 'entities'

flashMessages ViewHelper - renders the flash messages (if there are any)

<f:flashMessages renderMode="div" />	
renderMode	One of 'div' or 'ul'

for ViewHelper - foreach function

<pre><f:for each="{fruit1: 'apple', fruit2: 'pear', fruit3: 'banana', fruit4: 'cherry'}" as="fruit" key="label"> {label}: {fruit} </f:for></pre>			
<pre><f:for each="{0:1, 1:2, 2:3, 3:4}" as="foo" iteration="fooIterator"> Index: {fooIterator.index} Cycle: {fooIterator.cycle} Total: {fooIterator.total} {f:if(condition: fooIterator.isEven, then: 'Even')} {f:if(condition: fooIterator.isOdd, then: 'Odd')} {f:if(condition: fooIterator.isFirst, then: 'First')} {f:if(condition: fooIterator.isLast, then: 'Last')} </f:for></pre>			
each	Array of values or objects which is used for iteration	key	Key of iteration variable
as	Name of iteration variable	reverse	If TRUE, direction will be reversed
iteration	The name of the variable to store iteration information index, cycle, isFirst, isLast, isEven, isOdd, total		

groupedFor ViewHelper - grouping of results

<pre><f:groupedFor each="{0: {name: 'cherry', color: 'red'}, 1: {name: 'banana', color: 'yellow'}, 2: {name: 'strawberry', color: 'red'}}" as="fruitsOfThisColor" groupBy="color" groupKey="color"> {color} fruits: <f:for each="{fruitsOfThisColor}" as="fruit" key="label">{label}: {fruit.name}</f:for> </f:groupedFor></pre>			
as	Name of iteration variable	groupBy	Group by this property
each	Array or object which is used for iteration	groupKey	Name of variable which stores the grouping

if/then/else ViewHelper - if-then-else (w/o then if there is no else)

<pre><f:if condition="somecondition"><f:then>...</f:then><f:else>...</f:else></f:if></pre>	
<pre>Shorthand-Syntax: {f:if(condition: '{rank} > 100', then: 'rank is > 100', else: 'rank is <= 100')}</pre>	
condition	XX Comparator YY (e.g. <f:if condition="{rank} > 100">) Comparator is one of: ==, !=, <, <=, >, >= and % The % operator converts the result of the % operation to boolean. XX and YY can be one of: number / Object Accessor / Array / a ViewHelper Note: Strings at XX,YY are NOT allowed.

image ViewHelper - displays an image

<pre><f:image src="EXT:myext/Resources/Public/typo3_logo.png" alt="alt text" /> {f:image(src: 'EXT:myext/Resources/Public/logo.png', alt: 'alt text', minWidth: 30, maxWidth: 40)}</pre>			
alt	Specifies an alternate text for an image	width	Width of the image ('m' / 'c' possible)
ismap	Specifies an image as a server-side image-map.	height	Height of the image ('m' / 'c' possible)
longdesc	Specifies the URL to a document	minWidth	Minimum width of the image
usemap	Specifies an image as a client-side image-map	minHeight	Minimum height of the image
src	Source of the image	maxWidth	Maximum width of the image
		maxHeight	Maximum height of the image

form ViewHelper - form generation

absolute action	Render absolute action URI Form action (default is current url)	name	Name of form
actionUri	Override the "action" attribute	noCache	Disable caching (Bool)
additionalParams	Additional action URI query parameters	noCacheHash	Suppress the cHash (Bool)
addQueryString	Query params will be kept in the URI	pluginName	Plugin called
arguments	Additional arguments	pageUid	UID of target page
controller	Controller which should be called	pageType	Type of target page (default is 0)
enctype	MIME type for transmission	object	Object bound to the form - can be addressed through property
extensionName	Extension which should be called	objectName	Name of the object bound to the form (Default is name of form)
format	The requested format like "html"	onreset	JavaScript handler
		onsubmit	
fieldNamePrefix	Prefix that will be added to all field names within this form. If not set: tx_yourExtension_plugin	section	The anchor to be added to the URI
		argumentsToBeExcludedFromQueryString	Arguments to be removed from the action URI
method	Transfer method (default is "post")		

GENERAL ATTRIBUTES FOR ELEMENTS

errorClass	CSS class to set if there are errors	value	Value of element
name	Name of element	disabled	Displays the element disabled
property	Property of object which is bound through form		

form.checkbox - displays a checkbox

<pre><f:form.checkbox name="myCheckBox" value="someValue" checked="{object.value} == 5" /></pre>			
checked	If TRUE then checkbox is checked	property="myProperty" value="myValue"	if {object.myProperty} == myValue => checked

form.errors - displays the form errors (access {error.code} / {error.message} / {error.propertyName} / {error.errors})

<pre><f:form.errors for="someProperty"> {error.propertyName} / {error.code}: <f:for each="{error.errors}" as="errorDetail">{errorDetail.message}</f:for> </f:form.errors></pre>			
for	The errors of the given element	as	Name of error variable ('error')

form.hidden - displays a hidden field

<pre><f:form.hidden name="myHiddenValue" value="42" /></pre>	
--	--

form.password - displays a password input field

<pre><f:form.password name="myPassword" /></pre>			
maxLength	Maximum length of field	size	Length of input field
readonly	Readonly attribute of field		

form.radio - displays a radio button

<pre><f:form.radio name="myRadioButton" value="someValue" checked="{object.value} == 5" /></pre>		
checked	If TRUE then radiobutton is checked	see form.checkbox above (property="...")

form.select - displays a selector box

<pre><f:form.select name="users" options="{userArray}" optionValueField="id" optionLabelField="firstName" /></pre>			
selectAllByDefault	Select all options	options	Assoc array or object used for options
optionLabelField	Values for label are filled from property	size	Length of selector box
optionValueField	Values for value are filled from property	multiple	Display a multi-select box
sortByOptionLabel	List will be sorted by label		

form.submit - displays a submit button

<pre><f:form.submit value="Send Mail" /></pre>	
--	--

form.textarea - displays a text area

<pre><f:form.textarea name="myTextArea" value="This is shown inside the textarea" rows="5" cols="30" /></pre>			
rows	Number of rows	cols	Number of cols

form.textfield - displays an input field (formerly known as form.textbox)

<pre><f:form.textfield name="myTextBox" value="default value" /></pre>			
maxLength	Maximum length of field	required	Required attribute of field
placeholder	Placeholder attribute of field	size	Length of input field
readonly	Readonly attribute of field		

form.upload - displays an upload field (just works with enctype="multipart/form-data" in form tag)

<pre><f:form.upload name="file" /></pre>	
--	--

TYP03 Fluid Cheat Sheet 2

GIT: [git clone http://git.typo3.org/FLOW3/Packages/Fluid.git](http://git.typo3.org/FLOW3/Packages/Fluid.git)
 Forge: <http://forge.typo3.org/projects/show/package-fluid>
 Issue-Tracker: <http://forge.typo3.org/projects/show/v3v4-mvc/issues>



v 2.05 / 21.02.2012
Patrick Lobacher
www.typovision.de

typovision://

format ViewHelper - formats in different ways

format.crop - crops text			
<code><f:format.crop maxCharacters="17" append="&nbsp;[more]">This is some very long text</f:format.crop></code>			
append	String, which is appended at crop position	respectHtml	Cropped string will respect HTML
maxCharacters	Max. characters which are displayed	respectWordBoundaries	Crops only at word boundaries
format.currency - displays currency conversions			
<code><f:format.currency currencySign="\$" decimalSeparator="." thousandsSeparator=",">54321</f:format.currency></code>			
currencySign	The currency sign (like \$, €, ...)	thousandsSeparator	Character for thousands sep.
decimalSeparator	Character for decimal separation		
format.date - displays dates			
<code><f:format.date format="Hi">{dateObj}</f:format.date><f:format.date format="d.m.Y - Hi:s">{@timestamp}</f:format.date></code>			
format	Format in date() syntax.	date	Date/Time object or string
format.html - renders strings with TYPO3 parseFunc			
<code><f:format.html parseFuncSPATH="lib.parseFunc">foo bar. Some <LINK 1>link</LINK></f:format.html></code>			
parseFuncSPATH	Path to own parseFunc. Default is "lib.parseFunc RTE"		
format.nl2br - wrapper for PHP function nl2br			
<code><f:format.nl2br>{text_with_linebreaks}</f:format.nl2br> / {text_with_linebreaks -> f:format.nl2br() }</code>			
format.number - formats numbers			
<code><f:format.number decimals="1" decimalSeparator="," thousandsSeparator=".">423423.234</f:format.number></code>			
decimals	Numbers after comma. Default is "2"	thousandsSeparator	Character for thousands sep.
decimalSeparator	Character for decimal separation		
format.padding - wrapper for PHP function str_pad()			
<code><f:format.padding padLength="10" padString="-">TYPO3</f:format.padding></code>			
padLength	Length of output string	padString	String which is used for filling up
format.printf - wrapper for PHP function printf()			
<code><f:format.printf arguments="{number: 362525200}">% .3e</f:format.printf></code>			
<code><f:format.printf arguments="{0: 3, 1: 'Kasper'}">%25s is great, TYPO%15d too.</f:format.printf></code>			
arguments	Arguments for printf as array		

security ViewHelper

security.ifAuthenticated - implements an ifAuthenticated/else condition for FE users/groups			
<code><f:security.ifAuthenticated><f:then>Access.</f:then><f:else>No access.</f:else></f:security.ifAuthenticated></code>			
security.ifHasRole - implements an ifHasRole/else condition for FE users/groups.			
<code><f:security.ifHasRole role="Administrator"><f:then>You have the role.</f:then><f:else>You do not have the role.</f:else></f:security.ifHasRole></code>			
role	Group role (either UID or title)		

translate ViewHelper - (from Resources/Private/Language/locallang.xml)

<code><f:translate key="LLL:EXT:myext/Resources/Private/Language/locallang.xml:key1" /></code>			
<code>[f:translate(key: 'argumentsKey', arguments: {0: 'dog', 1: 'fox'}, default: 'default value')]</code>			
arguments	Arguments	htmlEscape	If FALSE, the output will not be escaped (TRUE)
default	Default key if 'key' is not found	key	Key in locallang file

link ViewHelper - generates links (link.xxx with tag / uri.xxx without tag)

name	Specifies the name of an anchor	target	Target parameter
rel	Rel: current => linked document	rev	Rel: linked => current document
link.action / uri.action - generates extbase action links			
<code><f:link.action action="show">action link</f:link.action></code>			
absolute	Render absolute URI	format	The format, e.g. ".html"
action	Target action	linkAccess	Show even access restricted pages
additionalParams	Additional parameters	noCache	Deactivate cache for target page
addQueryString	Query params kept in the URI	noCacheHash	No cHash parameter
arguments	Arguments	pageType	Page type (default 0)
argumentsToBeExcluded	Arguments to be removed from the action URI	pageUid	Target page UID (default current)
controller	Target Controller	pluginName	Target Plugin
extensionName	Target Extension	target	Target parameter

link.email - generates an email link

`<f:link.email email="foo@bar.tld">some custom content</f:link.email>`

email	Email address
-------	---------------

link.external - generates links to external targets

`<f:link.external uri="http://www.typo3.org" target="." blank">external link</f:link.external>`

defaultScheme	Scheme - 'http' is default	uri	Target URL
---------------	----------------------------	-----	------------

link.page - generates links to TYPO3 pages

`<f:link.page pageUid="1" additionalParams="{extension_key: 'foo: 'bar'}">page link</f:link.page>`

absolute	Render absolute URI	pageType	Page type (default 0)
additionalParams	Additional parameters	pageUid	Target page UID (default current)
addQueryString	Query params kept in the URI	page	Target page
argumentsToBeExcluded	Arguments to be removed from the action URI	linkAccess	Show even access restricted pages
FromQueryString		RestrictedPages	
section	Anchor	target	Target parameter
noCache	Deactivate cache for target page	uri	Target URL
noCacheHash	No cHash parameter		

be ViewHelper - backend module viewhelper

be.container - View helper which allows you to create extbase based modules in the style of TYPO3 default modules.

`<f:be.container pageTitle="foo" enableJumpToUrl="false" enableClickMenu="false" loadPrototype="false" loadScriptaculous="false" loadExtJs="true" loadExtJsTheme="false" extJsAdapter="jQuery" addCssFile="{f:uri.resource(path:'styles/backend.css')}> addJsFile="{f:uri.resource('scripts/main.js')}> your module content</f:be.container>`

addCssFile	Custom CSS file to be loaded	loadExtJs	Specifies whether to load ExtJS
addJsFile	Custom JavaScript file to be loaded	loadExtJsTheme	Whether to load ExtJS_grey theme
enableClickMenu	If TRUE (default), loads clickmenu.js	loadPrototype	Specifies whether to load prototype lib
enableExtJsDebug	If TRUE, debug version of ExtJS is loaded.	loadScriptaculous	Specifies whether to load scriptaculous
enableJumpToUrl	If TRUE (default), includes "jumpToUrl"	pageTitle	Title tag of the module
extJsAdapter	Load alternative adapter (ext-base is def.)	scriptaculousModule	Add. modules for scriptaculous

be.pageInfo - return page info icon

be.pagePath - current page path, prefixed with „Path.“ and wrapped in a span with the class "typo3-docheader-pagePath"

be.tableList - renders a record list as known from the TYPO3 list module

`<f:be.tableList tableName="fe_users" fieldList="{0: 'name', 1: 'email'}" storagePid="1" levels="2" filter="foo" recordsPerPage="10" sortField="name" enableClickMenu="false" clickTitleMode="info" alternateBackgroundColors="true" />`

alternate	Rows will have alternate background colors	readOnly	If TRUE, the edit icons won't be shown
BackgroundColors	"edit", "show" or "info"	recordsPerPage	Amount of records to be displayed
clickTitleMode	Enables context menu	sortDescending	Records sorted in descending order
enableClickMenu	Enables context menu	sortField	Table field to sort the results by
fieldList	List of fields to be displayed	storagePid	Records are fetched from this PID
filter	Corresponds to „Search String“ textbox	tableName	Name of the database table
levels	Corresponds to the level selector		

be.buttons.csh - CSH button as known from the TYPO3 backend

`<f:be.buttons.csh table="xMOD_csh_corebe" field="someCshKey" iconOnly="1" styleAttributes="border: 1px solid red" />`

iconOnly	If set, full text will never be shown	styleAttributes	Additional style-attribute
field	Field name (CSH locallang main key)	table	Table name ('_MOD_'+module)

be.buttons.icon - returns button with icon

`<f:be.buttons.icon uri="{f:uri.action('new')}> icon="new_el" title="Create new Foo" />`

add, add_workspace, button_down, button_hide, button_left, button_unhide, button_right, button_up, clear_cache, clip_copy, clip_cut, clip_pasteafter, closedok, datepicker, deletedok, edit2, helpbubble, icon_fatalerror, icon_note, icon_ok, icon_warning, new_el, options, perm, refresh_n, saveandclosedok, savedok, savedoknew, savedokshow, viewdok, zoom

icon	One of list above	uri	Target URI
title	Title attribute		

be.buttons.shortcut - returns shortcut button as known from the TYPO3 backend.

`<f:be.buttons.shortcut getVars="{0: 'M', 1: 'myOwnPrefix'}" setVars="{0: 'function'}" />`

getVars	List of GET variables to store	setVars	List of SET[] variables to store
---------	--------------------------------	---------	----------------------------------

be.menus.actionMenu / be.menus.actionMenuitem - select box, used to switch between multiple actions and controllers

`<f:be.menus.actionMenu><f:be.menus.actionMenuitem label="List Posts" controller="Post" action="index" arguments="{blog: blog}" /></f:be.menus.actionMenu>`

action	Action to be called	controller	Controller to be called
arguments	Arguments	label	Label of option tag

be.security.ifAuthenticated - implements an ifAuthenticated/else condition for BE users/groups (like security.ifAuthenticated)

be.security.ifHasRole - implements an ifHasRole/else condition for BE users/groups. (like security.ifHasRole)

API: Write own ViewHelper with name [vname]

Put file [Vname]ViewHelper.php in Classes/ViewHelpers/ with the following class definition:

class Tx_ExtKey_ViewHelpers_ [Vname]ViewHelper extends Tx_Fluid_Core_ViewHelper_AbstractViewHelper	Associative array of variables of viewhelper. Accessible within render()
\$arguments	Contains all variables which are accessible in the template. Change with add() & remove()
\$templateVariableContainer	Context of controller - serves as API for the controller. Supports the following properties: request, response, arguments, argumentsMappingResults, uriBuilder, FlashMessageContainer
\$controllerContext	Container for exchanging data between viewhelpers. Add with add() & read with get()
\$viewHelperVariableContainer	Register arguments within this method with \$this->registerArgument(\$name, \$type, \$description, \$required, \$defaultValue)
initializeArguments()	Code before rendering of viewhelper
initialize()	Render method which is responsible for output of viewhelper. It's required.
render(\$arg1, \$arg2, ...)	Called within render() will return all rendered content from child elements
renderChildren()	

class Tx_ExtKey_ViewHelpers_ [Vname]ViewHelper extends Tx_Fluid_Core_ViewHelper_AbstractTagBasedViewHelper	Name of Tag which will be generated by the viewhelper
\$tagName	Instance TagBuilder (API see below)
\$this->tag	Constructor. You should call parent::construct().
__construct()	Code before rendering of viewhelper. You should call parent::initialize()
initialize()	Registers a viewhelper argument which is directly used as tag attribute. Should be called within initializeArguments()
registerTagAttribute(\$name, \$type, \$description, \$required)	Registers all universal tag attributes like class, dir, id, lang, style, title, accesskey, ...
registerUniversalTagAttributes()	
relevant API of TagBuilder	
setContent(\$tagContent), getContent(), forceClosingTag(\$forceClosingTag), addAttribute(\$attributeName, \$attributeValue, \$escapeSpecialCharacters = TRUE), addAttributes(array \$attributes, \$escapeSpecialCharacters = TRUE), removeAttribute(\$attributeName), reset(), render()	

widget ViewHelper

widget.autocomplete - autocomplete for fields (needs Static TS 'Fluid: Default AJAX configuration')

`<f:widget.autocomplete for="name" objects="{posts}" searchProperty="author">`

for	Name of field for autocomplete	searchProperty	Property of object to search in
objects	Objects to search in		

widget.link / widget.uri - create links to extbase actions within widgets (link with a-tag / uri without)

`<f:widget.link action="show">link</f:widget.link>`

[action.xxx]	Same arguments as action.link or uri	ajax	TRUE for link to ajax widget
--------------	--------------------------------------	------	------------------------------

widget.paginate - renders a pagination of objects

`<f:widget.paginate objects="{blogs}" as="paginatedBlogs" configuration="{itemsPerPage: 3, insertAbove: 1, insertBelow: 1}">`

`<f:for each="{paginatedBlogs}" as="blog"><blog.title></f:for>`

`</f:widget.paginate>`

as	Identifier of paginated objects	objects	Objects that will be paginated
configuration	itemsPerPage, insertAbove, insertBelow		

API: Write own Widget ViewHelper with name [widget.vname]

Put ViewHelper-Class into Classes/ViewHelpers/Widget/[VhName]ViewHelper.php:

class Tx_ExtName_ViewHelpers_Widget_ [VhName]ViewHelper extends Tx_Fluid_Core_Widget_AbstractWidgetViewHelper {

* @var Tx_ExtName_ViewHelpers_Widget_Controller_ [VhName]Controller protected \$controller;

* @param Tx_ExtName_ViewHelpers_Widget_Controller_ [VhName]Controller \$controller public function injectController(Tx_ExtName_ViewHelpers_Widget_Controller_ [VhName]Controller \$controller) { \$this->controller = \$controller;

public function render() { return \$this->initiateSubRequest(); }

Put Widget-Controller into Classes/ViewHelpers/Widget/Controller/[VhName]Controller.php:

class Tx_ExtName_ViewHelpers_Widget_Controller_ [VhName]Controller extends Tx_Fluid_Core_Widget_AbstractWidgetController {

initializeAction() {...}

[actionName]Action() {...}

Put Templates in Resources/Private/Templates/ViewHelpers/Widget/[VhName]/[ActionName].html