TemplaVoila!

Ключевое слово расширения: **templavoila**

Оглавление

TemplaVoila!.....1
Введение.....3
О чем это?.....3
Спонсорство.....3
О документе.....3
Инструкция разработчика сайта.....3
Справочник по конфигурированию Page / User TSconfig.....3
mod.web_txtemplavoilaM1.....3
mod.web_txtemplavoilaM2.....5
Атрибуты плагина “pi1”.....5
Руководство по программированию расширения. 5
Кто будет это читать?.....5
Пользовательские функции/ Обработчики (User functions / hooks).....6
Добавление пунктов к боковому меню(sidebar).....7
TemplaVoila API для разработчиков расширений...7
Расширения <T3DataStructure>.....8
Введение.....8
Расширения <T3DataStructure> для “<tx_templavoila>”.....8
Страницы и TemplaVoila.....12
Доступ к “родительской(parent)” записи из DS TypoScript.....12
Соображения: Шаблоны API для применения в приложениях (plugins).....12
Введение.....12
Выбор альтернативных Шаблонов(Template Object).....12
Опции Расширения – Гибкие формы(FlexForm).....12
Создание Объекта Шаблона(Template Object).....13
Установка файла Data Structure XML для преобразования Template Object.....16
Получение значения поля из Plugin FlexForm.....18
Обнаружение записи Template Object (TO).....18
Объединение данных с помощью разметки шаблона(Template markup).....19
FAQ: Часто задаваемые вопросы.....21
Подстраницы не наследуют datastructure / template object.....21

Введение

О чем это?

Если кратко, то TemplaVoila является альтернативным движком шаблонов предлагающим новый подход при создании и работе с элементами дизайна. На самой вершине обеспечивается новый интерфейс пользователя во внутреннем интерфейсе (aka “Модуль страницы(Page Module)”).

Расширение “TemplaVoila” было разработано Kasper Skårhøj и Robert Lemke в качестве проекта для большой французской компании Dassault Systemes. TemplaVoila явилась результатом нововведений, которые решали некоторые проблемы, возникшие в проекте. В частности, TemplaVoila предназначена для создания более гибкой структуры страниц, чем существующая в TYPO3 концепция “колонок”. В дальнейшем, она интегрирует традиционное создание шаблонов на уровне элементов контента, с более удобным чем прежде стилем point-n-click. Наконец, разработка TemplaVoila также сопровождается некоторыми расширениями ядра TYPO3, в особенности, концепции называемой гибгие формы(FlexForms), которая позволяет из внутреннего интерфейса TYPO3 создавать иерархические формы и сохранять их содержимое в структурах XML.

Спонсорство

Основная часть этого расширения была щедро профинансирована фирмой Dassault Systémes, France. Огромная благодарность за ваше невероятное отношение к Open Source и очень серьезную поддержку TYPO3. Завершение версии 1.0 было профинансировано TYPO3 Association. Спасибо всем нашим спонсорам, которые способствовали нашей разработке!

О документе

Несмотря на релиз TemplaVoila 1.0, руководство все еще до конца не окончено. Но это не означает, что нет документации по TemplaVoila ... Следующие документы смогут помочь вам начать и обеспечат важной информацией для работы:

- Руководство TemplaVoila (этот документ) – Этот документ всегда содержит законченный справочник доступных опций конфигурации. Пока он не содержит будущую инфомацию об установке сайтов TemplaVoila, обновлении и миграции с сайтов не на TemplaVoila. Но, по-крайней мере, части обновления и миграции планируются.
- Futuristic Template Building – Это большое пошаговое руководство, которое разъясняет важнейшие моменты вTemplaVoila и позволяет начать первый сайт на TV. На момент написания, часть информации руководства требует обновления для отражения новых возможностей , которые появились за последний год. Но все еще стоит пытаться получить обзор!
- Руководство по локализации – Документ с подробным объяснением локализации и переводу в TYPO3 в общих чертах. В нем специально приводятся возможности локализации / перевода TemplaVoila. На момент написания (07.04.06) этот документ пока не опубликован на TYPO3.org, но уже очень скоро.

Инструкция разработчика сайта

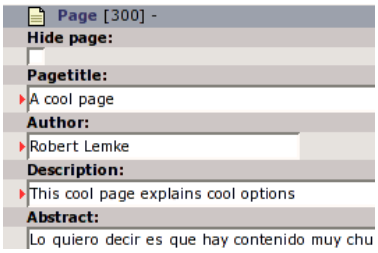
Предназначается для разработчиков TYPO3 сайтов. Знание PHP не требуется.

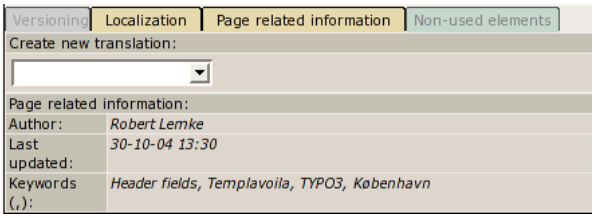
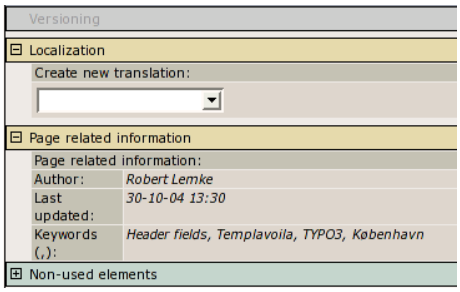
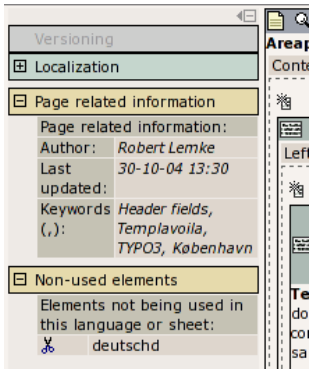
Справочник по конфигурированию Page / User TSconfig

Следующая конфигурация TypoScript может применяться как Page TSconfig или как User TSconfig.

mod.web_txtemplavoilaM1

Свойство:	Тип:	Описание:	Значение по умолчанию:

createPageWizard.fieldNames	list	С помощью этой опции вы можете определить выбор имен полей таблицы *страниц* для отображения в помощнике создания-новой-страницы(create-new-page-wizard).
		
		Пример:
		`mod.web_txtemplavoilaM1.createPageWizard {`
		`fieldNames = hidden, title, author, description, abstract`
		`}`
		
		Здесь создается форма редактирования вроде этой:
		 
		
			hidden, title, alias

Свойство:	Тип:	Описание:	Значение по умолчанию:
sideBarEnable	boolean	Определяет видимость или нет панели инструментов в модуле страниц TemplaVoila(Page Module). По-умолчанию, она видна как строка закладок.	TRUE
sideBarPosition	string	Определяет положение панели инструментов в TemplaVoila (Page Module). Возможны значения: *toptabs, toprows, left* ***toptabs*** генерирует панель инструментов наверху контента модуля страницы мод с динамиченскими закладками, которые позволяют переключаться между различными категориями опций.  ***toprows*** также генерирует панель инструментов наверху страницы, но применяет строки, которые сворачиваются или разворачиваются вместо закладок.  ***left*** создается слева сбоку в модуле страницы. Эта боковая панель может показываться или скрываться при клике плюса / минуса верхнего правого(?)(upper right corner) угла панели.  **Примечание:** Пункты меню, которые доступны в этой панели зависят от установленных расширений, поскольку они обеспечивают функциональность.	toptabs
disableContainerElementLocalization-Warning	boolean	Контейнерные элементы применяемые в TemplaVoila не будут локализованы. Поэтому, появится сообщение об ошибке, если <langDisable> является false для таких структур данных. Если же локализация запланирована, то это сообщение конечно вводит в заблуждение, и его можно отключить этой установкой.	FALSE
disableContainerElementLocalization-Warning_warningOnly	boolean	Иногда вам может потребоваться локализовать элементы контейнерас помощью <langChildren>. Это специальный случай, для которого элемент является не только контейнером, но и имеет поля контента требующие локализации. Задача состоит в том, чтобы язык по-умолчанию полей ссылок (не полей контента((non-content fields)) распознавался модулем страниц TemplaVoila, в то время как обработка для внешнего интерфейса зависит от наследования и от других ссылок, которые могут случайно поместить поля ссылок на другие языки! Итак, если у вас локализация с такой смесью записей, то а) убедитесь, что наследование включено (чтобы для всех языков использовалась ссылка на язык по-умолчанию) и b) не существует локализованных ссылок (оставьте поля ссылок на другие языки пустыми).	FALSE

Свойство:	Тип:	Описание:	Значение по умолчанию:
translationParadigm	string keyword	Если установлено в “free”, то модуль страниц будет действать согласно парадигме перевода называемой “Free” (в противоположность “Bound”), для которой применяется Page DS <langDisable> для индикации возможности локализации страницы или ее отсутствия и что существующие записи локализации языка по-умолчанию указывают на различные структуры контента из структур данных либо по наследованию либо раздельно. Вам следует прочесть документ “Localization Guide” в котором детально рассмотрена эта концепция.	
disableDisplayMode	list of keywords	В парадигме перевода “Bound” вы увидите блок выбора, который позволяет вам фильтровать видимые языки при редактировании. Здесь можно отключить определенную из доступных опций, добавляя ключевое слово в этот список. Доступные опции: default, selectedLanguage, onlyLocalized	

[tsconfig:mod.web_txtemplavoilaM1]

mod.web_txtemplavoilaM2

Свойство:	Тип:	Описание:	Значение по умолчанию:
templatePath	string	Путь к каталогу внутри fileadmin/ где можно найти шаблоны. Должен быть доступным с помощью монтирования пользовательских файлов. **Пример:** mod.web_txtemplavoilaM2.templatePath = template/main/	

[tsconfig:mod.web_txtemplavoilaM2]

Атрибуты плагина “pi1”

Свойство:	Тип:	Описание:	Значение по умолчанию:
TSconst.[constant-name]	string	Определяет константы для Data Structure constants. Смотри раздел “расширения <T3Data Struction>”	
dontInheritValueFromDefault			
childTemplate	string	Ключевое слово применяемое для просмотра записи шаблона-наследника. По-умолчанию вы можете задать значение “print”. Оно также устанавливается автоматически при обнаружении в URL “&print=1”. Значение сопоставляется с контентом “rendertype”, таким образом, если вы захотите задать значения отличные от “print”, то вы просто добавите новые пункты в этот блок выбора и примените условия в TypoScript Template, для активации условия, которое установит это значение. Подробности позднее	
disableExplosivePreview	boolean	Если установлено, то выпадающие информационные блоки не будут появляться во внешнем (FE) режиме предпросмотра.	FALSE
disableErrorMessages	boolean	Если установлено, то сообщения об ошибках не появятся во внешнем интерфейсе. **Пример:** `plugin.tx_templavoila_pi1.disableErrorMessages = 1`	FALSE

[tsref:plugins.tx_templavoila_pi1]

Руководство по программированию расширения

Кто будет это читать?

Если вы программируете расширения и хотите понять как использовать определенные части TemplaVoila или повлиять на обработку в собственных расширениях, тогда этот раздел руководства для вас. Требуется знание PHP, в такой же мере как и некоторое умение в программировании расширений TYPO3.

Пользовательские функции/ Обработчики (User functions / hooks)

Там где важно, мы внедрили некоторые обработчики, где была возможность, в TemplaVoila, чтобы позволить при программировании расширений их переписать или добавить определенной функциональностити. Просто зарегистрируйте вашу собственную функцию и вы получите управление или влияние на часть TemplaVoila.

Если вам необходимо расширить определенную часть и вы не видите такой возможности, просто сообщите нам, мы, возможно, вставить какие-нубудь API для подключения пользовательских функций.

Вообще-то, существует два пути для обеспечения обратчиков, те – что используют t3lib_div::getUserObj() и те – что используют t3lib_div::callUserFunction(). Для обработчиков использующих ***getUserObj*** **требуется class name** , в то время как обработчики ***callUserFunction*** принимают **class name и method name**. Ниже приведен пример регистрации вашей функции для обоих случаев:

```
   // The getUserObject way:
$TYPO3_CONF_VARS['EXTCONF']['templavoila'][ sub_key ][subsub_key][] = 'my_class';

   // The callUserFunction way:
$TYPO3_CONF_VARS['EXTCONF']['templavoila'][ sub_key ][subsub_key][] = 'my_class->my_method';
```

Какой из типов обработчиков выполняется указано в колонке типа(*type) в таблице ниже*. Там же указано одна или множество пользовательских функций разрешено для этого обработчика. В последнем случае, вам следует добавить ваш class name (и method) к **массиву userfunctions**.

Совет: Вам следует познакомиться с разделом об обработчиках в документе *TYPO3 core API*, который доступен на TYPO3.org. И, конечно, вам следует взглянуть на исходный текст в котором применен обработчик перед созданием своей пользовательской функции.

Подключ:	Под-подключ:	Тип:	Назначение / Описание:

cm1	eTypesConfGen	callUserFunction / single	“eTypes” является предопределенным набором, применяемым в модуле click module (cm1) для создания конфигурации полей в структуре данных. Во время отбражения вы можете выбирать между этими eTypes, например “text”, “image”, “imagefixed”, “ce” и т.д.
			Например, “input” eType, создает следующую конфигурацию внутри структуры данных:
			
			`<TCEforms>`
			`  <config>`
			`  <type>input</type>`
			`  <size>30</size>`
			`  <eval>trim</eval>`
			`  </config>`
			`  <label>test</label>`
			</TCEforms>
			
			Если вам требуется переопределить создание этой конфигурации для определенного eType, то вы можете применить eTypesConfGenUserfunctions для определения пользовательской функции.
			
			Для обеспечения пользовательской функции для eType “input”, вы можете определить нечто вроде этого на странице TSconfig в вашем расширении:
			
			`$TYPO3_CONF_VARS['EXTCONF']['templavoila']['cm1']['eTypesConfGen']['input'] = 'tx_myClass->myMethod'`
			
			Создание вашей пользовательской функции детально представлено в templavoila/cm1/index.php

cm1	eTypesExtraFormFields	callUserFunction / single	(Также рассмотрите изложение выше об eTypes) Используя этот обработчик вы можете определить функцию пользователя, которая будет обрабатывать определенные экстра поля(extra fields) для определенных eTypes в диалогах преобразований. Одним из популярных экстра полей является путь к объекту для TypoScriptObject eType. **Пример:** `$TYPO3_CONF_VARS['EXTCONF']['templavoila']['cm1']['eTypesExtraFormFields']['input'] = 'tx_myClass->myMethod';`
mod1	renderTopToolbar	callUserFunction / multiple	Используйте этот обработчик если есть необходимость вывода некоторого кода HTML на самом верху экрана редактирования страницы в модуле страницы. Этот обработчик был внедрен для обеспечения пользовательской панели инструментов связанной с текущей страницей. **Пример:** `$TYPO3_CONF_VARS['EXTCONF']['templavoila']['mod1']['renderTopToolbar'][] = 'tx_myClass->myMethod';`

Подключ:	Под-подключ:	Тип:	Назначение / Описание:
mod1	renderPreviewContent	getUserObj / multiple	Используйте этот обработчик если вам нужно обработать предпросмотр пользовательский cType или переопределить предпросмотр по-умолчанию определенного типа cType. Это просто прекрасно, когда вы можете предусмотреть предпросмотр для своих программ! Давайте договоримся, что написали свою программу с именем myext_pi1. Просто создайте ваш класс новой функции и зарегистрируйте его в $TYPO3_CONF_VARS (смотри выше). Ваша функция будет примерно такой: **Пример:** `function renderPreviewContent_preProcess ($row, $table, &$alreadyRendered, &$reference) { if (row['CType'] == 'list' && $row['list_type'] == 'myext_pi1') { $content = 'MyExt: '.htmlspecialchars('my custom preview'); $alreadyRendered = true; return $reference->linkEdit($content, $table, $row['uid']); } }`
mod1	renderFrameworkClass	getUserObj / multiple	Эта функция содержит следующий обработчик: **renderFrameWork_preProcessOutput** Вызывается перед окончательным отображением контента кадра(framework) на экране страницы редактирования.Примером использования этого обработчика является добавление другой иконки в строку заголовка определенного элемента контента или страницы.
pi1	renderElementClass	getUserObj / multiple	Эта функция содержит следующий обработчик: **renderElement_preProcessRow** Дает вам возможность модифицировать строку, только что подготовленную для отображения во внешнем интерфейсе. Одним из способов применения является выбор различных объектов шаблона для универсальных (flexible) элементов контента, основанных на условиях.

Добавление пунктов к боковому меню(sidebar)

[TODO: Объяснить каким образом другие расширения могут легко добавить пункты к боковому меню]

```
if (t3lib_extMgm::isLoaded('templavoila'))    {
    require_once (t3lib_extMgm::extPath('templavoila').'mod1/class.tx_templavoila_mod1_sidebar.php');
}
class tx_myext_templavoila_sidebar {
    function init() {
            // Create / get instances:
        $thisObj =& t3lib_div::getUserObj ('&tx_myext_templavoila_sidebar', '');
        $sideBarObj =& t3lib_div::getUserObj ('&tx_templavoila_mod1_sidebar', '');
            // Register sidebar item:
        $sideBarObj->addItem ('tx_myext_templavoila_sidebar_item1', $thisObj, 'renderItem_myext', 'My
Extension', 50);
    }
    function renderItem_myext(&$pObj) {
            // Dummy output, just return the current page id:
        return $pObj->id;
    }
}



if (t3lib_extMgm::isLoaded('templavoila'))    {
    require_once (t3lib_extMgm::extPath($_EXTKEY).'class.tx_myext_templavoila_sidebar.php');
    tx_myext_templavoila_sidebar::init();
}
```

TemplaVoila API для разработчиков расширений

- Основная концепция
- flexformPointer / flexformPointerString
  - sourcePointer / destinationPointer

Расширения <T3DataStructure>

Введение

TemplaVoila расширяет Data Structure XML набором тегов, которые определяют две вещи связанные с TemplaVoila:

- **Преобразование(Mapping):** Определение правил преобразования, описания, пример данных и типы предустановленных полей
- **Обработка(Rendering):** Определение кода TypoScript, Object Path, флаги обработки и константы

Расширения <T3DataStructure> для “<tx_templavoila>”

Элементы “Массив(Array)”:

Элемент	Описание	Под-элементы
<[application tag]>	В этом случае меткой приложения является “<tx_templavoila>”	<title> <description> <tags> <sample_data> <sample_order> <eType> <TypoScriptObjPath> <TypoScript> <proc> <ruleConstants> <ruleRegEx> <ruleDefaultElements> <langOverlayMode>
<ROOT><tx_templavoila>	Для элементов <ROOT> в DS	<title> <description> <pageModule>
<pageModule>	Набор опций конфигурации, которые влияют на обработку в модуле страниц.	<displayHeaderFields> <titleBarColor>
<sample_data>	Пример данных, определенный в числовом массиве. Примерные данные выбираются случайно из этих опций	<n[0-x]>
<sample_order>	Для <section>: Определяет набор объектов массивов для отображения в примере данных. Каждое значение является числовым массивом указывающим на имя поля в массиве объекта <el>.	<n[0-x]>
<TypoScript_constants>		<[constant_name]>
<proc>	Опции выполнения (во время обработки)	<stdWrap> <int> <HSC>

Элементы “Значение(Value)”:

Элемент	Формат	Описание
<meta><sheetSelector>	string	Определение с помощью кода PHP file/class для оценки выбора страницы во внешнем (frontend) интерфейсе. Это ссылка getUserObject вроде “EXT:user_myext/class.user_myext_selectsheet.php:&user_myext_selectsheet” в которой класс user_myext_selectsheet содержит функцию, selectSheet(), возвращающую ключ страницы, например “sDEF” для страницы по-умолчанию. **Заметка об использовании шаблонов при обработке во внешнем интерфейсе (pi1):** Эта возможность достаточно сложна и все еще нуждается в некоторой доработке и документировании. Отметим следующее: ● Когда страницы определены, тогда шаблон также требуется переопределить! ● Если не существует определение для остальных ключей кроме “sDEF”, тогда они по-умолчанию используют определение “sDEF”. Таким образом, экономится определение раз за разом, если все страницы используют один и тот же шаблон. ● При использовании страниц локальная обработка XML, также нуждается в применении контейнера, например “<sheet><sDEF> .... </sheet></sDEF>” ● Выбор страниц следует производить тщательно, только тех которые определяются параметрами сохраненными в кэш. Это достигается для параметров известных как cHash protected – или при выключенном кэш страницы, конечно.
<title>	string	Заголовок отображаемый при выводе преобразования(mapping view)

Элемент	Формат	Описание
<description>	string	Инструкции/описание пребразования, показываемое при выводе преобразования(mapping view).
<tags>	string	Список правил тегов через запятую. Тег правила определяется как [tagname]:[mapping-mode]:[attribute] **Примеры:** • table:outer,div,body:inner,td:inner • *:attr:href • a:attr:* • *:inner,a:attr:href,a:attr:src
<eType>	string	Значения, указывающие на предустановки TCEforms. Применяется для создания структур данных(Data Structures) в templavoila. Устанавливаются и управляются автоматически. Этот тег используется внутри инструментом преобразования.
<oldStyleColumnNumber>	integer	Установка этого тега равным целому значению (обычно между 0 и 3), вы определяете к какому номеру колонки это поле относится. Эта информация используется модулем списка(list module), редактированием внешнего интерфейса и во всех остальных местах, которые работают со старой парадигмой. Если вы хотите конвертировать до-TemplaVoila сайт в сайт TemplaVoila с помощью помощника миграции, то вы также должны убедиться что теги oldStyleColumnNumber установлены для ваших областей контента. **Примечание:** Каждое значение может использоваться лишь однажды в в структуре данных и это использование влияет только на страничные шаблоны! **Вводная информация:** Перед TemplaVoila, контент страницы группировался с помощью назначения для каждого элемента контента id определенной колонки. По-умолчанию, были доступны четыре колонки: “Normal” (id=0), “Left” (id=1), “Right” (id=2) и “Border” (id=3). Некоторые части TYPO3 и некоторые расширения безразличны к тому, что TemplaVoila структурирует контент иначе. Если вы создаете или перемещаете элемент контента в модуле списка(List module), то возможно элемент не появится там где вы ожидаете, поскольку модульсписка не знает какая область контента является колонкой “Normal”. **Пример:** <T3DataStructure> <ROOT> <el> <field_maincontent> <tx_templavoila> <oldStyleColumnNumber>0</oldStyleColumnNumber> ...
<TypoScriptObjPath>	string	Путь объекта TypoScript указывающий на TypoScript Template Content Object, который будет обрабатывать контент представленный элементом. Это очень полезно, если вы хотите вставить меню, которое определено как, например,. “lib.myMenu” в TypoScript Template сайта.

Элемент	Формат	Описание

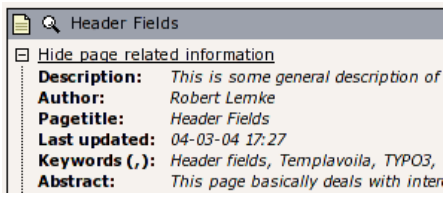
<TypoScript>	string	Контент TypoScript.
		
		Можно вставить константы
		• которые определены локально в <TypoScript_constants>, смотри ниже
		• В шаблоне TypoScript на сайте; в поле Setup вы можете установить константы как свойства (только первого уровня) в “plugin.tx_templavoila_pi1.TSconst” - те которые можно вставить с помощью {$TSconst.[constant name]} в данные <TypoScript>!
		
		Основной Пример:
		
		`<TypoScript>`
		`<![CDATA[`
		
		`10 = USER`
		`10.userFunc = user_3dsplm_pi2->testtest`
		`10.imageConfig {`
		`file.import.current = 1`
		`file.width = 100`
		`}`
		
		`]]>`
		`</TypoScript>`
		
		Доступ к другим полям в той же структуре данных:
		
		`<TypoScript>`
		`10 = TEXT`
		`10.field = field_myotherfield`
		`</TypoScript>`
		
		Отобразить заголовок страницы:
		
		`<TypoScript>`
		`10 = TEXT`
		`10.data = page:title`
		`</TypoScript>`

<[constant_name]>	string	Локальная константа TypoScript, которую можно вставить с помощью {$[constant_name]} в <TypoScript> (смотри выше)
		
		Вместо заданию плоского значения вы также можете ссылаться на значения ссылок объектов из шаблонов TypoScript сайтов, включая значение вроде “{$lib.myConstant}”. **Отметим**, что значение находится в поле установки шаблонов(Templates Setup).
		
		Пример:
		`<TypoScript_constants>`
		`<backGroundColor>red</backGroundColor>`
		`<fontFile>{$_CONSTANTS.resources.fontFile}</fontFile>`
		`</TypoScript_constants>`
		
		Здесь “_CONSTANTS.resources.fontFile” должно быть частью объекта с значением определенным в шаблоне TypoScript на сайте!

<int>	boolean, 0/1	Обработать в intval() перед выводом
<HSC>	boolean, 0/1	Обработать в htmlspecialchars() перед выводом

<stdWrap>	string	Свойства StdWrap как TypoScript, например:
		
		`<proc>`
		`<stdWrap>`
		`trim = 1`
		`br = 1`
		`</stdWrap>`
		`</proc>`

Элемент	Формат	Описание
<langOverlayMode>	string, keyword	Установка изменения режима для контента при использовании <meta><langChildren> и других языков в гибких формах (flexforms). Обычно наследование языка по-умолчанию всегда активно и глобально может выключаться с помощью установки в TypoScript “dontInheritValueFromDefault” при необходимости. Тем не менее, в Data Structure и TO / Local Processing XML вы можете это пересмотреть для каждого(per-field) поля с помощью этого ключа. В любом случае, это оказывает влияние только на значения отличные от языка по-умолчанию и только если активно <langChildren> (таким образом, применение “vDEF” и родственных полей с именами “vXXX” для локализации). **Ключевые слова:** **ifFalse –** Контент наследует, если он оценен как false в PHP (это значит, что строки нуль и пробел(zero и blank) приводят к изменению) **ifBlank** - Контент наследует, если он отмечен как пустая строка(усечен((trimmed)). **never** - Контент никогда не наследует от языка по-умолчанию! **removeIfBlank** – Если значение этого поля пустое, тогда *вся группа* полей(элементов) удаляется! Это способ для удаления единичного элемента для локализации при конструкции <langChildren>=1 вместо наследования контента из языка по-умолчанию. **[default]** – Если не используется ключевых слов, то применяется глобальный режим.

<displayHeaderFields>	string	Список странично зависимых полей, который будет отображаться в качестве заголовка при редактировании страницы в модуле страниц. Пока разрешена только таблица “page” / чтобы почувствовать.
		
		Примечание: Этот тег влияет только на секции верхнего уровня <tx_templavoila>, т.е. один уровень ниже корневого тега <ROOT>.
		
		 
		
		Пример:
		`<T3DataStructure>`
		`  <ROOT>`
		`  <tx_templavoila>`
		`  <pageModule>`
		`  <displayHeaderFields>`
		`  pages.keywords`
		`  pages.mycustomfield`
		`  </displayHeaderFields>`
		`  </pageModule>`
		`...`

<titleBarColor>	color	Если вы захотите помочь своим редакторам при определении структур данных страниц, над которыми они работают, вы можете определить цвет используя этот тег. Блок заголовка на самом верху экрана редактируемой страницы будет отображаться этим цветом.
		
		Вы можете воспользоваться любым цветом, разрешенным в CSS (например “red”, также как и “#FC2300” и т.п.)
		
		Примечание: Этот тег влияет только на секции верхнего уровня <tx_templavoila>, т.е. один уровень ниже корневого тега <ROOT>.
		
		 
		
		Пример:
		`<T3DataStructure>`
		`  <ROOT>`
		`  <tx_templavoila>`
		`  <pageModule>`
		`  <titleBarColor>orange</titleBarColor>`
		`...`

Расширения тегов в Data Structure		

Элемент	Формат	Описание
<[field-name]><type>	string	В Data Structure чуствительны только “array” или пробел(blank). Тем не менее, для TemplaVoila возможны дополнительные значения, “attr” и “no_map”. Это является завершением обзора в TemplaVoila <type> / <section> означающим: • <type>array</type> = Обрабатывается массив или объекты • <type>array</type> + <section>1</section> = Обрабатывается секция которая должна содержать другой тип массива(array-type) (без установки <section>!) • <type>attr</type> = Объект помечен как *атрибут* тега HTML. • <type>[blank]</type> = Объект помечен как *элемент* тега HTML. • <type>no_map</type> = Объект не отображаемый (только для редактирования в FlexForms к примеру.)

Страницы и TemplaVoila

TemplaVoila является совместимой с определением странц. В таком контексте, страница с элементом <ROOT> показывается в структуре отображения содержащей каждую страницу как элементы <ROOT> под ней.

Доступ к “родительской(parent)” записи из DS TypoScript

Примечание: Эта возможность является экспериментальной и может быть изменена в будущем. Все пользователи предупреждены: пользуйтесь на свой страх и риск! Это примечание будет опущено, когда эта функция станет стабильной.

Для доступа к “родительской(parent)” записи из таблиц “tt_content” или “pages” в секции поля <TypoScript>, разработчик может воспользоваться специальными регистрами. Эти регистры определены только при выполнении секции <TypoScript>. В примере ниже показано как использовать эти регистры:

```
<TypoScript>
10 = TEXT
10.data = register:tx_templavoila_pi1.parentRec.uid
10.wrap = “uid” field of parent record is |
</TypoScript>
```

Таким образом, любое поле родительской записи определяется как регистр **tx_templavoila_pi1.parentRec.XXX**, где XXX заменяется именем поля из соответсвующей таблицы.

Отметим, что эти регистры не определены для статических структур данных, поскольку у статических структур данных отсутствуют ассоциированные родительские записи. Если появится ссылка **tx_templavoila_pi1.parentRec.XXX** на статическую структуру данных, то результат непредсказуем.

Соображения: Шаблоны API для применения в приложениях (plugins)

Введение

Вы можете воспользоваться TemplaVoilas API для шаблонов в ваших собственных приложениях при желании. В качестве примера, давайте рассмотрим как работает расширение “mininews”:

Расширение mininews имеет три вывода контента:

- Список всех новостей в архиве, включая блок поиска и ссылки для перехода на следующую страницу, если новостей оказалось больше 20.
- Детальное отображение единичной
- Краткое представление во внешнем интерфейсе списка последних новостей со ссылками “read more”.

Каждое из этих отбражений по-умолчанию обрабатываются как запрограммированный HTML в расширении. Запрограммированный HTML создан таким образом, чтобы быть полезным в большинстве случаев, так как вы можете сами подобрать стили отображения в CSS. Иными словами, вам может оказаться незачем создавать альтернативный шаблон!

Таким образом, если у вас появится желание изменить вывод значительно, больше чем вам может позволить CSS с HTML по-умолчанию, вы можете создать шаблон TemplaVoila. Расширение mininews поддерживает это.

Выбор альтернативных Шаблонов(Template Object)

Опции Расширения – Гибкие формы(FlexForm)

Альтернативный шаблон выбирается в опциях расширения как объект шаблона, которые обрабатываются как стандартные элементы гибких форм для расширений:

Для заинтересовавшихся, эта форма генерируется с помощью конфигурации “flexform_ds.xml” в расширении “mininews”:

```
<T3DataStructure>
    <meta>
        <langDisable>1</langDisable>
    </meta>
    <ROOT>
        <type>array</type>
        <el>
            <field_templateObject>
                <TCEforms>
                    <label>LLL:EXT:mininews/locallang_db.php:tt_content.pi_flexform.select_template</label>
                    <config>
                        <type>select</type>
                        <items>
                            <n0>
                                <n0></n0>
                                <n1>0</n1>
                            </n0>
                        </items>
                        <foreign_table>tx_templavoila_tmplobj</foreign_table>
                        <foreign_table_where>
                            AND tx_templavoila_tmplobj.pid=###STORAGE_PID###
                            AND tx_templavoila_tmplobj.datastructure="EXT:mininews/template_datastructure.xml"
                            AND tx_templavoila_tmplobj.parent=0
                            ORDER BY tx_templavoila_tmplobj.title
                        </foreign_table_where>
                        <size>1</size>
                        <minitems>0</minitems>
                        <maxitems>1</maxitems>
                    </config>
                </TCEforms>
            </field_templateObject>
        </el>
    </ROOT>
</T3DataStructure>
```

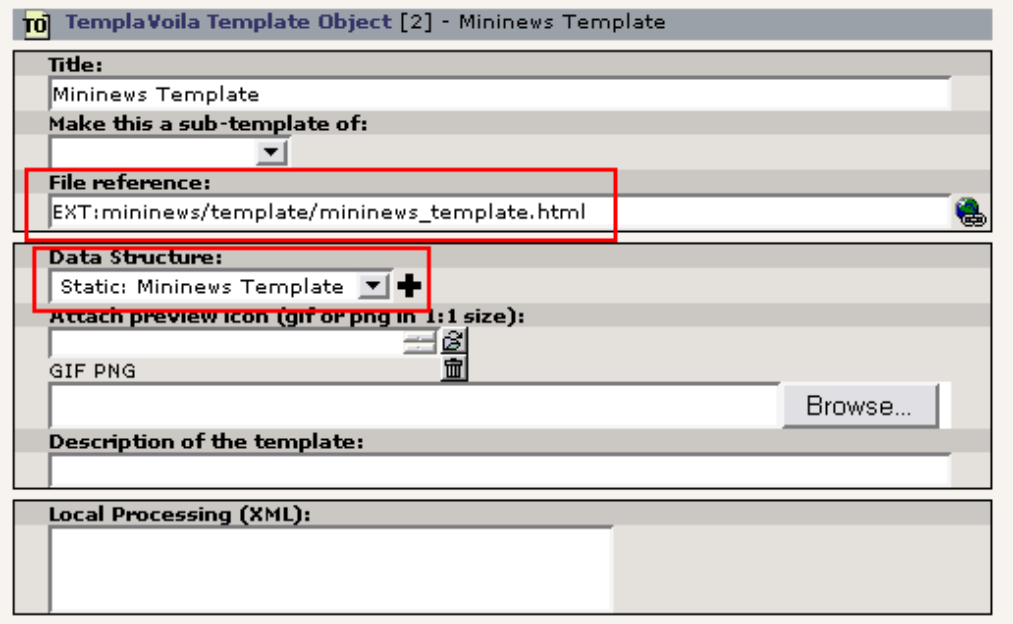
В дальнейшем “mininews” активизирует конфигурацию установкой этих строк в “ext_tables.php”

```
$TCA['tt_content']['types']['list']['subtypes_addlist'][$_EXTKEY.'_pi1']='tx_mininews_frontpage_list;;;;
1-1-1,pi_flexform';
t3lib_extMgm::addPiFlexFormValue($_EXTKEY.'_pi1', 'FILE:EXT:mininews/flexform_ds.xml');
```

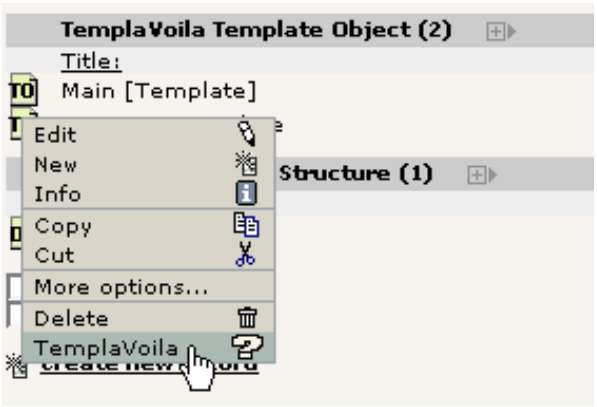
Это все, что вам позволено для выбора альтернативного шаблона. Конечно остается вопрос – что такое Template Object и как нам его создать? Ответ найдется ниже.

Создание Объекта Шаблона(Template Object)

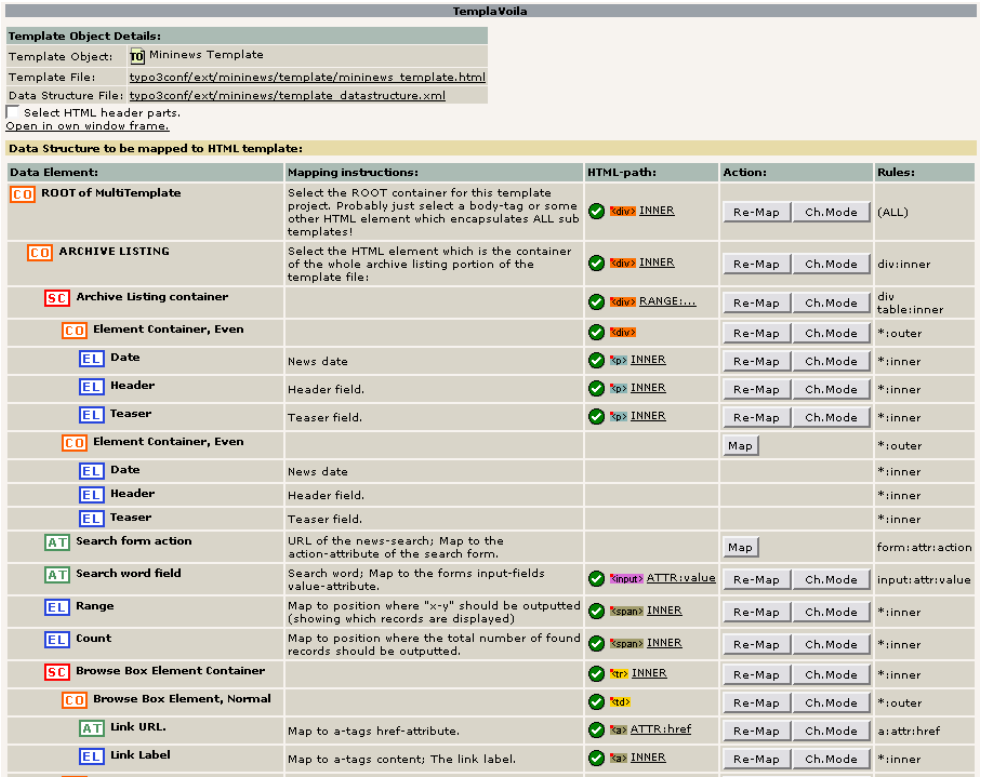
Это делается в “Storage folder”, который конфигурируется для сайта. Здесь вы создаете новый Template Object и выбираете *Data Structure,* которые расширение mininews создает:



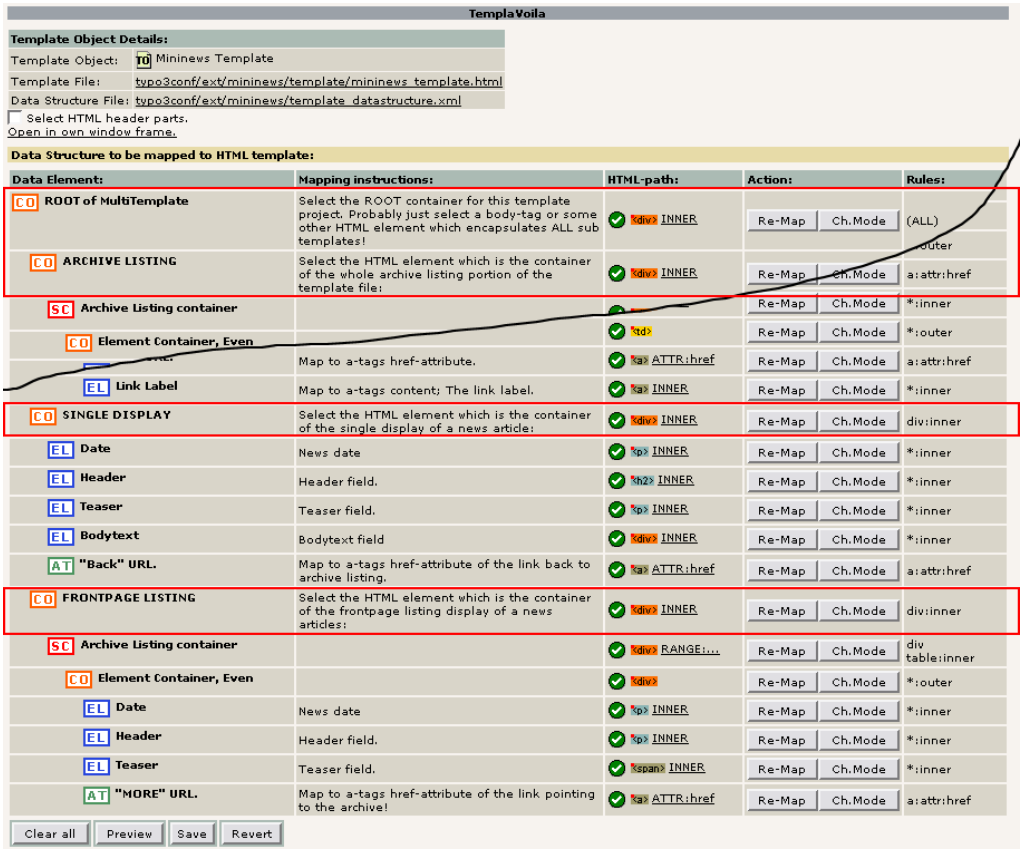
После выполнения этого, документ закрывается, кликаем снова иконку Template Object и выбираем “TemplaVoila”:



Позднее вы можете начать преобразовывать Data Structure в файл шаблона (здесь применяется пример такого файла “mininews/template/mininews_template.html”) и после завешения вы увидите нечто похожее на это:



Отметим, в частности, как каждый из трех шаблонов присутствует в *той же* Data Structure как *страницы* для которых наивысший корневой элемент называемый “ROOT of multitemplate” представляет внутри три страницы:



После окончания процесса отображения получен альтернативный шаблон.

Теперь большими вопросами являются:

- Каким образом я могу определить структуры данных для собственного расширения также как в “mininews”?
- Каким образом я могу использовать альтернативный шаблон представленный Template Object внутри моего расширения?

Ответ на эти вопросы последует.

Установка файла Data Structure XML для преобразования Template Object

Структура данных “mininews” которая используется для отображения шаблонов помещена в файл “mininews/template_datastructure.xml”. Содержимое выглядит так:

```
<T3DataStructure>
   <sheets>
   <!-- The Archive configuration is so large that we have put it into it's own file,
         and references it from here: -->
      <sArchive>EXT:mininews/template_datastructure_arc.xml</sArchive>

      <!-- Single display of mininews items: -->
      <sSingle>
         <ROOT>
            <tx_templavoila>
               <title>SINGLE DISPLAY</title>
                  <description>Select the HTML element which is the container of the
                  single display of a news article:</description>
                   <tags>div:inner</tags>
            </tx_templavoila>
            <type>array</type>
            <el>
               <field_date>
                  <tx_templavoila>
                     <title>Date</title>
                     <description>News date</description>
                     <tags>*:inner</tags>
                     <sample_data>
                        <n0>6th August 10:34</n0>
                        <n1>29/12 2003</n1>
                     </sample_data>
                  </tx_templavoila>
               </field_date>
               <field_header>
                  <tx_templavoila>
                     <title>Header</title>
                     <description>Header field.</description>
                        <tags>*:inner</tags>
                        <sample_data>
                           <n0>People on mars!</n0>
                           <n1>Snow in Sydney</n1>
                        </sample_data>
                  </tx_templavoila>
               </field_header>
               <field_teaser>
                  <tx_templavoila>
                     <title>Teaser</title>
                     <description>Teaser field.</description>
                     <tags>*:inner</tags>
                     <sample_data>
                        <n0>Capthurim Chanaan vero genuit Sidonem primogenitum et
                              Heth Iebuseum quoque </n0>
                     </sample_data>
                  </tx_templavoila>
               </field_teaser>
               <field_bodytext>
                  <tx_templavoila>
                     <title>Bodytext</title>
                        <description>Bodytext field</description>
                        <tags>*:inner</tags>
                        <sample_data>
                           <n0><![CDATA[
                              <p><strong>Filii Ham Chus et Mesraim Phut et Chanaan</strong> filii autem
Chus Saba et Evila Sabatha et Rechma et Sabathaca porro filii Rechma Saba et Dadan Chus autem genuit
Nemrod iste coepit esse potens in terra Mesraim vero genuit Ludim et Anamim et Laabim et Nepthuim
Phethrosim quoque et Chasluim de quibus egressi sunt Philisthim et.</p>
                              <p>Capthurim Chanaan vero genuit Sidonem primogenitum et Heth Iebuseum
quoque et Amorreum et Gergeseum Evheumque et Aruceum et Asineum Aradium quoque et Samareum et Ematheum
filii Sem Aelam et Assur et Arfaxad et Lud et Aram et Us et Hul et Gothor et Mosoch Arfaxad autem genuit
Sala qui et ipse genuit Heber porro Heber nati sunt duo filii nomen uni Phaleg quia in diebus eius
divisa est terra et nomen fratris eius Iectan Iectan autem genuit Helmodad et Saleph et Asermoth et Iare
Aduram quoque et Uzal et Decla Ebal etiam et Abimahel et Saba necnon et Ophir et Evila et Iobab omnes
isti filii Iectan Sem Arfaxad Sale.</p>
                           ]]></n0>
                        </sample_data>
                     </tx_templavoila>
                  </field_bodytext>
                  <field_url>
                     <type>attr</type>
```

```
            <tx_templavoila>
               <title>"Back" URL.</title>
               <description>Map to a-tags href-attribute of the link back to
                  archive listing.</description>
               <tags>a:attr:href</tags>
               <sample_data>
                  <n0>javascript:alert('You click this link!');</n0>
               </sample_data>
            </tx_templavoila>
         </field_url>
      </el>
   </ROOT>
</sSingle>


<!-- Frontpage display of a few mininews teaser items: -->
<sFrontpage>
   <ROOT>
      <tx_templavoila>
         <title>FRONTPAGE LISTING</title>
         <description>Select the HTML element which is the container of the
                     frontpage listing display of a news articles:</description>
         <tags>div:inner</tags>
      </tx_templavoila>
      <type>array</type>
      <el>
         <field_fpListing>
            <type>array</type>
            <section>1</section>
            <tx_templavoila>
               <title>Archive Listing container</title>
               <description></description>
               <tags>div,table:inner</tags>
            </tx_templavoila>
            <el>
               <element_even>
                  <type>array</type>
                  <tx_templavoila>
                     <title>Element Container, Even</title>
                     <description></description>
                     <tags>*:outer</tags>
                  </tx_templavoila>
                  <el>
                  <field_date>
                     <tx_templavoila>
                        <title>Date</title>
                        <description>News date</description>
                        <tags>*:inner</tags>
                        <sample_data>
                           <n0>6th August 10:34</n0>
                           <n1>29/12 2003</n1>
                        </sample_data>
                     </tx_templavoila>
                  </field_date>
                  <field_header>
                     <tx_templavoila>
                        <title>Header</title>
                        <description>Header field.</description>
                        <tags>*:inner</tags>
                        <sample_data>
                           <n0>People on mars!</n0>
                           <n1>Snow in Sydney</n1>
                        </sample_data>
                     </tx_templavoila>
                  </field_header>
                  <field_teaser>
                     <tx_templavoila>
                        <title>Teaser</title>
                        <description>Teaser field.</description>
                        <tags>*:inner</tags>
                        <sample_data>
                           <n0>Capthurim Chanaan vero genuit Sidonem primogenitum et
                               Heth Iebuseum quoque </n0>
                        </sample_data>
                     </tx_templavoila>
                  </field_teaser>
                  <field_url>
                     <type>attr</type>
                     <tx_templavoila>
                        <title>"MORE" URL.</title>
                        <description>Map to a-tags href-attribute of the link pointing
                              to the archive!</description>
                        <tags>a:attr:href</tags>
```

```
                                    <sample_data>
                                       <n0>javascript:alert('You click this link!');</n0>
                                    </sample_data>
                                 </tx_templavoila>
                              </field_url>
                           </el>
                        </element_even>
                     </el>
                  </field_fpListing>
               </el>
            </ROOT>
         </sFrontpage>
      </sheets>
   </T3DataStructure>
```

Если изучать эту распечатку, то можно обнаружить, что в ней конфигурируется только DS для шаблонов “SINGLE DISPLAY” и “FRONTPAGE LISTING” - “ARCHIVE LISTING” фактически находится в другом файле, в соответствии с этой строкой:

```
<sArchive>EXT:mininews/template_datastructure_arc.xml</sArchive>
```

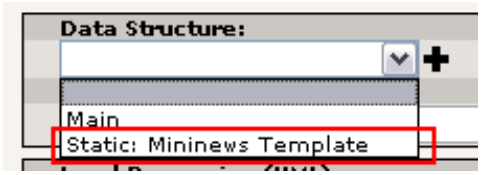
Вы можете изучить его содержимое самостоятельно.

Как бы то ни было, вопрос остается: Каким образом mininews конфигурируется чтобы этот файл XML был доступен для Template Objects для указания? Это располагается в файле ext_tables.php:

```
// Adding datastructure for Mininews:
$GLOBALS['TBE_MODULES_EXT']['xMOD_tx_templavoila_cm1']['staticDataStructures'][]=array(
    'title' => 'Mininews Template',
    'path' => 'EXT:'.$_EXTKEY.'/template_datastructure.xml',
    'icon' => '',
    'scope' => 0,
);
```

$_EXTKEY содержит значение “mininews”, как обычно, в файле ext_tables.php для расширения.

С помощью такой конфигурации DS появится в блоке выбора Data Structure:



На этот момент мы имеем:

- Конфигурацию FlexForm с необходимостью выбора записи шаблона в “Insert Plugin” типа Content Element
- Структуру данных(Data Structure (DS)) в файле XML, которая используется для отображения файла шаблона HTML в DS.
- У вас также есть пример файла шаблона HTML, которые демонстрирует отображение в вашу DS.

Все указанное применимо для применения шаблона в расширении.

Получение значения поля из Plugin FlexForm

В классе расширения mininews нам сначала нужно определить есть ли ссылки на запись Template Object и если есть, то быть уверенным, что они используются.

Обнаружение записи Template Object (TO)

В файле “mininews/pi1/class.tx_mininews_pi1.php” класс состоит из двух переменных:

```
        // TemplaVoila specific:
    var $TA='';                     // If TemplaVoila is used and a TO record is found, this array will
be loaded with Template Array.
    var $TMPLobj='';            // Template Object
```

Позднее, в функции просмотра списка(listView) вы обнаружите эту инициализацию, которая обнаружит запись. Комментарии ниже

```
   1: function listView($content,$conf)    {
   2:
   3:         // Init FlexForm configuration for plugin:
   4:     $this->pi_initPIflexForm();
   5:
   6:         // Looking for TemplaVoila TO record and if found, initialize template object:
   7:     if (t3lib_extMgm::isLoaded('templavoila'))    {
   8:         $field_templateObject = $this->pi_getFFvalue($this->cObj-
>data['pi_flexform'],'field_templateObject');
   9:         if (intval($field_templateObject))    {
  10:             $this->TMPLobj = t3lib_div::makeInstance('tx_templavoila_htmlmarkup');
  11:             $this->TA = $this->TMPLobj->getTemplateArrayForTO(intval($field_templateObject));
  12:             if (is_array($this->TA))    {
  13:                 $this->TMPLobj->setHeaderBodyParts($this->TMPLobj->tDat['MappingInfo_head'],$this-
>TMPLobj->tDat['MappingData_head_cached']);
  14:             }
  15:         }
  16:     }
```

- Строка 4 инициализирует поле в “pi_flexform” $this->cObj->data. Этим конвертируется поле из строки с данными XML в массив с тем же XML конвертированным в массив PHP с помощью t3lib_div::xml2array()
- Строка 7 проверяет загрузку TemplaVoila – что просто необходимо!
- Строка 8 запрашивает значение “field_templateObject” из контента FlexForm “pi_flexform”
- Строка 9 проверяет является ли значение целым – что означает указание на запись Template Object - “uid”
- В строке 10 мы создаем экземпляр класса “tx_templavoila_htmlmarkup” которй будет нашим API для объединения наших *данных* из mininews с *шаблоном* из записи Template Object.
- Строка 11 загружает Массив Шаблона(Template Array) из TO по указателю из $field_templateObject.
- Строка 12 проверяет установку Массива Шаблона(Template Array) – что имеет место, если существует информация преобразования в TO.
- Строка 13 будет устанавливать возможные секции й заголовков, если какой-нибудь будет определен в TO.

Наконец, оставшийся класс mininews мы можем просто проверить, если $this->TA является массивом, и если это так, то -применим templavoilas API для объединения данных и шаблона. Это продемонстрировано ниже.

Объединение данных с помощью разметки шаблона(Template markup)

С тем чтобы не удлинять (видимо изложение), я просто вырежу некоторые примеры.

Строки с повторяющимися списками

В первом примере показано каким образом аккумулировать контент для списка строк. Это обычно делается в цикле, перемещаясь по элементам и в каждой такой итерации вызывается функция API в TemplaVoila с двумя аргументами, соответствующая часть переменной ->TA (Template Array = cached template markup) и массив с данными mininews.

```
   1:     // Create list of elements:
   2: $elements='';
   3: while($this->internal['currentRow'] = mysql_fetch_assoc($res))    {
   4:     $elements.=$this->TMPLobj->mergeDataArrayToTemplateArray(
   5:         $this->TA['sub']['sArchive']['sub']['field_archiveListing']['sub']['element_even'],
   6:         array(
   7:             'field_date' => $this->getFieldContent('datetime'),
   8:             'field_header' => $this->pi_list_linkSingle($this->getFieldContent('title'),$this-
>internal['currentRow']['uid'],1),
   9:             'field_teaser' => nl2br(trim(t3lib_div::fixed_lgd($this-
>getFieldContent('teaser_list'),$this->conf['frontPage.']['teaserLgd'])))
  10:         )
  11:     );
  12: }
```

В этой распечатке вы можете обнаружить, что массив данных из mininews имеет три ключа, “field_date”, “field_header” и “field_teaser”. Они связаны с тремя элементами из Data Structure XML для шаблона “ARCHIVE LISTING”:

```
  10:                 <field_archiveListing>
  11:                     <type>array</type>
  12:                     <section>1</section>
  13:                     <tx_templavoila>
  14:                         <title>Archive Listing container</title>
  15:                         <description></description>
```

```
  16:                                     <tags>div,table:inner</tags>
  17:                                 </tx_templavoila>
  18:                                 <el>
  19:                                     <element_even>
  20:                                         <type>array</type>
  21:                                         <tx_templavoila>
  22:                                             <title>Element Container, Even</title>
  23:                                             <description></description>
  24:                                             <tags>*:outer</tags>
  25:                                         </tx_templavoila>
  26:                                         <el>
  27:                                             <field_date>
  28:                                                 <tx_templavoila>
  29:                                                     <title>Date</title>
  30:                                                     <description>News
date</description>
  31:                                                     <tags>*:inner</tags>
  32:                                                     <sample_data>
  33:                                                         <n0>6th August
10:34</n0>
  34:                                                         <n1>29/12 2003</n1>
  35:                                                     </sample_data>
  36:                                                 </tx_templavoila>
  37:                                             </field_date>
  38:                                             <field_header>
  39:                                                 <tx_templavoila>
  40:                                                     <title>Header</title>
  41:                                                     <description>Header
field.</description>
  42:                                                     <tags>*:inner</tags>
  43:                                                     <sample_data>
  44:                                                         <n0>People on mars!</n0>
  45:                                                         <n1>Snow in Sydney</n1>
  46:                                                     </sample_data>
  47:                                                 </tx_templavoila>
  48:                                             </field_header>
  49:                                             <field_teaser>
  50:                                                 <tx_templavoila>
  51:                                                     <title>Teaser</title>
  52:                                                     <description>Teaser
field.</description>
  53:                                                     <tags>*:inner</tags>
  54:                                                     <sample_data>
  55:                                                         <n0>Capthurim Chanaan
vero genuit Sidonem primogenitum et Heth Iebuseum quoque </n0>
  56:                                                     </sample_data>
  57:                                                 </tx_templavoila>
  58:                                             </field_teaser>
```

Это часть Data Structure, которая встроена внутри

```
<T3DataStructure><sheets><sArchive><ROOT><el>
```

Я специально показываю это, чтобы вы увидели логику переменной

$this->TA['sub']['sArchive']['sub']['field_archiveListing']['sub']['element_even']

в коде распечатки. По существу, если вы подставите “sub” с “el”, то вы также сможете увидеть что эта переменная будет содержать разметку для

```
<T3DataStructure><sheets><sArchive><ROOT><el><field_archiveListing><el><element_even>

$this->TA['sub']['sArchive']['sub']['field_archiveListing']['sub']['element_even']
```

А теперь все вместе

После завершения сбора списка строк (и кое-чего еще) значения на внешних уровнях также собраны с помощью соответствующего вызова API, чей вывод в конце концов возвращается:

```
   1:     // Wrap the elements in their containers:
   2: $out = $this->TMPLobj->mergeDataArrayToTemplateArray(
   3:         $this->TA['sub']['sArchive'],
   4:         array(
   5:             'field_archiveListing' => $elements,
   6:             'field_browseBox_cellsContainer' => $br_elements,
   7:             'field_searchBox_sword' => htmlspecialchars($this->piVars['sword']),
   8:             'field_searchBox_submitUrl' => htmlspecialchars(t3lib_div::getIndpEnv('REQUEST_URI')),
   9:             'field_browseBox_displayRange' => $rangeLabel,
  10:             'field_browseBox_displayCount' => $this->internal['res_count']
```

```
  11:         )
  12:     );
  13:
  14: return $out;
```

Теперь можно увидеть, что аккумулированный контент списка строк ($elements) добавлен к ключу “field_archiveListing”. Для всех остальных полей, которые можно найти также в DS:

```
...
 105:
 106:                   <!--
 107:                           Defining mappings for the search box:
 108:                   -->
 109:                   <field_searchBox_submitUrl>
 110:                           <type>attr</type>
 111:                           <tx_templavoila>
 112:                                   <title>Search form action</title>
 113:                                   <description>URL of the news-search; Map to the action-attribute
of the search form.</description>
 114:                                   <tags>form:attr:action</tags>
 115:                                   <sample_data>
 116:                                           <n0>javascript:alert('Hello, you pressed the search
button!');return false;</n0>
 117:                                   </sample_data>
 118:                           </tx_templavoila>
 119:                   </field_searchBox_submitUrl>
 120:                   <field_searchBox_sword>
 121:                           <type>attr</type>
 122:                           <tx_templavoila>
 123:                                   <title>Search word field</title>
 124:                                   <description>Search word; Map to the forms input-fields value-
attribute.</description>
 125:                                   <tags>input:attr:value</tags>
 126:                                   <sample_data>
 127:                                           <n0>Strawberry Jam</n0>
 128:                                           <n1>Jack Daniels</n1>
 129:                                           <n2>Flowers</n2>
 130:                                   </sample_data>
 131:                           </tx_templavoila>
 132:                   </field_searchBox_sword>
 133:
 134:                   <!--
 135:                           Defining mappings for the browse box, display note:
 136:                   -->
 137:                   <field_browseBox_displayRange>
 138:                           <tx_templavoila>
 139:                                   <title>Range</title>
 140:                                   <description>Map to position where "x-y" should be outputted
(showing which records are displayed)</description>
 141:                                   <tags>*:inner</tags>
 142:                                   <sample_data>
 143:                                           <n0>1-10</n0>
 144:                                           <n1>20 to 30</n1>
 145:                                   </sample_data>
 146:                           </tx_templavoila>
 147:                   </field_browseBox_displayRange>
 148:                   <field_browseBox_displayCount>
 149:                           <tx_templavoila>
 150:                                   <title>Count</title>
 151:                                   <description>Map to position where the total number of found
records should be outputted.</description>
 152:                                   <tags>*:inner</tags>
 153:                                   <sample_data>
 154:                                           <n0>123</n0>
 155:                                           <n1>3402</n1>
 156:                                   </sample_data>
 157:                           </tx_templavoila>
 158:                   </field_browseBox_displayCount>
...
```

Thats all!

FAQ: Часто задаваемые вопросы

Подстраницы не наследуют datastructure / template object

Q: *Во внешнем интерфейсе, страницы обрабатываются если я выбираю структуру данных(datastructure) и шаблон, но если я определяю некоторые DS/TO для подстраниц, то появляется сообщение об ошибке: Не обнаружена установка Data Structure*

для table/row XXX. Пожалуйста, сначала выберите Data Structure и Template Object.

A: Убедитесь, что вы правильно сконфигурировали расширение для внешнего интерфейса (templavoila_pi1) в вашем шаблоне TypoScript. Следующая конфигурация приведет к ошибке приведенной выше:

```
page = PAGE
page.typeNum = 0
page.10 < plugin.tx_templavoila_pi1
```

Вместо этого примените следующий код и просмотрите главу конфигурации в этом руководстве:

```
page = PAGE
page.typeNum = 0
page.10 = USER
page.10.userFunc = tx_templavoila_pi1->main_page
```

Q: *Я создал поле "Content Elements", добавил элементы контента, но ничего не выводится во внешнем интерфейсе(FrontEnd). Что делать?*

A: Вы должны проверить существование входа <TypoScriptObjectPath> в том поле внутри записи DS. Если он существует, TemplaVoila не покажет контента. Это проявляется в версиях TemplaVoila до 1.0 включительно, но будет исправлено в будущих версиях. Проверьте запись DS и удалите любое вхождение <TypoScriptObjectPath> из полей элементов контента. Не забудьте очистить кэш перед проверкой резулттатов.

Q: *Я сделал новый сайт, но во внешнем интерфейсе всегда пусто!*

A: Вы забыли добавить шаблоны из расширения "CSS static content" в записи шаблона.