EXT: Static Info Tables

Table of Contents

Copyright....................2
Credits....................2
Introduction....................3
What does it do?....................3
Requirements....................3
Support....................3
Static Tables....................4
General rules....................4
Read only....................4
UTF-8 encoding....................4
Non-removal of deleted entries....................4
Localization by language pack....................4
Standards....................4
Tables structure....................4
Territory (static_territories)....................4
Country (static_countries)....................5
Country Zone (static_country_zones)....................5
Currency (static_currencies)....................5
Language (static_languages)....................6
Tables extension by language pack....................6
Extbase domain model....................7
Objects, properties and methods....................7
Territory....................7
Country....................7
CountryZone....................8
Currency....................8
Language....................9
Domain model extension by language pack....................9
Additional properties....................9
Localization property nameLocalized....................9
Repositories and methods....................9
Abstract repository....................9
Territory....................10
Country....................10
CountryZone....................10
Currency....................10
Language....................10
Using the model....................11
Adding a country select field to a form....................11
Using the select View Helper....................11
Further examples....................11
Configuration of backend forms....................12
Localization of entity selector....................12
Alternate value fields for entity selector....................12
Localization of suggest wizard....................13
Pre-Extbase API....................14
API class....................14
Class initialization....................14
Getting a localized name....................14
Creating a localized drop-down selector....................14
Formating an amount....................14
Loading an alternate currency....................15
Formating an address....................15
Getting country info....................15
Creating/updating a language pack....................16
Creating a new language pack....................16
Updating an existing language pack....................16
Installation....................17
Installing the extension....................17
Upgrading to version 12.4.1+....................17
Upgrading to version 6.0+ from older versions....................17
Configuration....................18
Extension configuration....................18
TypoScript reference....................18
Address formats....................19

Copyright

Credits

Thanks to René Fritz, Eckhard Zemp, David Brühlmeier, Franz Holzinger, Martin Kutschker and Carsten Biebricher for their contributions.

Introduction

What does it do?

Static Info Tables is a collection of database tables which provides data on:

- territories,
- countries,
- country zones (states, provinces, local government areas),
- languages,
- and currencies.

These tables are static. They should be used as reference only. The records can be found in the TYPO3 backend in the root level of the page tree.

The extension provides

- the tables definition and non-localized contents;
- an extensible Extbase domain model and generic repository access methods;
- an API for traditional pre-Extbase plugins and modules;
- an Extbase backend module for creating language packs: extensions which provide localization of the tables contents.

Requirements

Version 12.4.0 requires TYPO3 12 LTS.

Version 11.5.0 requires TYPO3 11 LTS.

Version 6.9.0 requires TYPO3 9 LTS and is enabled in TYPO3 10 LTS.

Version 6.6.0 requires TYPO3 8 LTS.

Version 6.4.0 requires TYPO3 7 LTS.

Version 6.0 requires TYPO3 6.0.6 and PHP 5.3.7.

Version 2.3.2 is the last version that will work with TYPO3 4.3-4.7 and PHP 5.2.

Support

Please report issues on : https://codeberg.org/sjbr/static-info-tables/issues

Static Tables

General rules

Read only

The tables are static, i.e. they are not intended to be updated on a site using the extension. Therefore, the tables are normally configured as read-only.

UTF-8 encoding

The tables are encoded with UTF-8.

Non-removal of deleted entries

In order to ensure that the value of the uid field of an entry is never changed and that relationships established between tables on the basis of the uid fields, entries marked as deleted should never be removed from the tables.

Localization by language pack

The tables provided by this extension do not include labels in any language other than English, except for some labels in the local language of the entity. Labels in other languages are provided by extra extensions, called language packs, that extend the basic strutures provided by this extension.

Standards

Standards relevant to the Static Info Tables are:

- for territories or macro-geographical regions: UN geoscheme (http://unstats.un.org/unsd/methods/m49/m49regin.htm);
- for country codes: ISO 3166-1 (see http://www.iso.org/iso/country_codes);
- for country subdivisions: ISO 3166-2;
- for currencies: ISO 4217;
- for languages: ISO 639-1 and RFC 4646 (see http://www.faqs.org/rfcs/rfc4646.html);
- for postal address formats: UPU S42-1 (see http://www.upu.int/en/activities/addressing/postal-addressing-systems-in-member-countries.html).

A major source of reference data is provided by the Locale Data Repository (CLDR) Project http://www.unicode.org/cldr/.

Tables structure

Territory (static_territories)

This table provide data about territories or macro-geographical regions as defined by the UN geoscheme (http://unstats.un.org/unsd/methods/m49/m49regin.htm).

The data is hierarchical organized which means the fields tr_parent_iso_nr and tr_parent_territory_uid refer to the parent territory containing the territory defined by the current entry.

A country has a reference to the territory in which the country is contained.

```
CREATE TABLE static_territories (
  uid int(11) unsigned NOT NULL auto_increment,
  pid int(11) unsigned NOT NULL default '0',
  deleted tinyint(4) NOT NULL default '0',
  tr_iso_nr int(11) unsigned NOT NULL default '0',
  tr_parent_territory_uid int(11) NOT NULL default '0',
  tr_parent_iso_nr int(11) unsigned NOT NULL default '0',
  tr_name_en varchar(50) NOT NULL default '',
  PRIMARY KEY (uid)
);
```

Country (static_countries)

This table provides data about countries based on ISO 3166-1 (see http://www.iso.org/iso/country_codes);

```
CREATE TABLE static_countries (
  uid int(11) unsigned NOT NULL auto_increment,
  pid int(11) unsigned NOT NULL default '0',
  deleted tinyint(4) NOT NULL default '0',
  cn_iso_2 char(2) NOT NULL default '',
  cn_iso_3 char(3) NOT NULL default '',
  cn_iso_nr int(11) unsigned NOT NULL default '0',
  cn_parent_territory_uid int(11) NOT NULL default '0',
  cn_parent_tr_iso_nr int(11) unsigned NOT NULL default '0',
  cn_official_name_local varchar(128) NOT NULL default '',
  cn_official_name_en varchar(128) NOT NULL default '',
  cn_capital varchar(45) NOT NULL default '',
  cn_tldomain char(2) NOT NULL default '',
  cn_currency_uid int(11) NOT NULL default '0',
  cn_currency_iso_3 char(3) NOT NULL default '',
  cn_currency_iso_nr int(10) unsigned NOT NULL default '0',
  cn_phone int(10) unsigned NOT NULL default '0',
  cn_eu_member tinyint(3) unsigned NOT NULL default '0',
  cn_uno_member tinyint(3) unsigned NOT NULL default '0',
  cn_address_format tinyint(3) unsigned NOT NULL default '0',
  cn_zone_flag tinyint(4) NOT NULL default '0',
  cn_short_local varchar(70) NOT NULL default '',
  cn_short_en varchar(50) NOT NULL default '',
  cn_country_zones int(11) NOT NULL default '0',
  PRIMARY KEY (uid)
);
```

Local names (cn_official_name_local and cn_short_local) are in the official languages of the countries. Appropriate Unicode fonts are required to display all entries on a client computer.

Address formats are described in the section Address Formats of the configuration section.

Country Zone (static_country_zones)

This table provides data about country subdivisions, provinces, departments or states based on ISO 3166-2.

```
CREATE TABLE static_country_zones (
  uid int(11) unsigned NOT NULL auto_increment,
  pid int(11) unsigned NOT NULL default '0',
  deleted tinyint(4) NOT NULL default '0',
  zn_country_iso_2 char(2) NOT NULL default '',
  zn_country_iso_3 char(3) NOT NULL default '',
  zn_country_iso_nr int(11) unsigned NOT NULL default '0',
  zn_code varchar(45) NOT NULL default '',
  zn_name_local varchar(128) NOT NULL default '',
  zn_name_en varchar(50) NOT NULL default '',
  zn_country_uid int(11) NOT NULL default '0',
  zn_country_table tinytext,
  PRIMARY KEY (uid)
);
```

Currency (static_currencies)

This table provides data about currencies based on standard ISO 4217.

```
CREATE TABLE static_currencies (
  uid int(11) unsigned NOT NULL auto_increment,
  pid int(11) unsigned NOT NULL default '0',
  deleted tinyint(4) NOT NULL default '0',
  cu_iso_3 char(3) NOT NULL default '',
  cu_iso_nr int(11) unsigned NOT NULL default '0',
  cu_name_en varchar(50) NOT NULL default '',
  cu_symbol_left varchar(12) NOT NULL default '',
  cu_symbol_right varchar(12) NOT NULL default '',
  cu_thousands_point char(1) NOT NULL default '',
  cu_decimal_point char(1) NOT NULL default '',
  cu_decimal_digits tinyint(3) unsigned NOT NULL default '0',
  cu_sub_name_en varchar(20) NOT NULL default '',
  cu_sub_divisor int(11) NOT NULL default '1',
  cu_sub_symbol_left varchar(12) NOT NULL default '',
  cu_sub_symbol_right varchar(12) NOT NULL default '',
  PRIMARY KEY (uid),
  KEY parent (pid)
);
```

Language (static_languages)

This table provides data about languages based on standards ISO 639-1 and RFC 4646 (see http://www.faqs.org/rfcs/rfc4646.html).

```
CREATE TABLE static_languages (
  uid int(11) unsigned NOT NULL auto_increment,
  pid int(11) unsigned NOT NULL default '0',
  deleted tinyint(4) NOT NULL default '0',
  lg_iso_2 char(2) NOT NULL default '',
  lg_name_local varchar(99) NOT NULL default '',
  lg_name_en varchar(50) NOT NULL default '',
  lg_typo3 char(2) NOT NULL default '',
  lg_country_iso_2 char(2) NOT NULL default '',
  lg_collate_locale varchar(5) NOT NULL default '',
  lg_sacred tinyint(3) unsigned NOT NULL default '0',
  lg_constructed tinyint(3) unsigned NOT NULL default '0',
  PRIMARY KEY (uid),
  KEY parent (pid)
);
```

The local name (lg_name_local) of a language is in the language itself. Appropriate Unicode fonts are required to display all entries on a client computer.

Tables extension by language pack

A language pack will add columns to these tables in the following way.

A language pack is identified by a two-letter ISO language code (e.g. de), possibly followed by an underscore and a two-letter ISO country code (e.g. de_AT).

Let ## stand for the lower-cased identifier of the language pack (e.g. de or de_at), then the following columns will be added to the tables:

- static_territories: tr_name_##
- static_countries: cn_short_##
- static_country_zones: zn_name_##
- static_currencies: cu_name_## and cu_sub_name_##
- static_languages: lg_name_##

Extbase domain model

Objects, properties and methods

The domain model provides the following objects onto which are mapped the corresponding static tables.

The domain model uses the name space SJBR\StaticInfoTables\Domain\Model

Territory

The class is named: SJBR\StaticInfoTables\Domain\Model\Territory

The following properties have a getter and a setter:

- deleted (whether or not the terrritory was removed from the UN standard)
- nameEn (English name)
- nameLocalized (localized name of the territory, non-persistent)
- parentTerritoryUnCodeNumber (UN code of containing territory)
- unCodeNumber (UN code)

Country

The class is named: SJBR\StaticInfoTables\Domain\Model\Country

The following properties have a getter and a setter:

- addressFormat
- capitalCity (name of captital city)
- countryZones (country subdivisions of the country)
- currencyIsoCodeA3
- currencyIsoCodeNumber
- deleted (whether or not the country was removed from the ISO standard)
- euMember (whether or not the country is a member of the European Union)
- additional method: isEuMember()
- isoCodeA2
- isoCodeA3
- isoCodeNumber
- nameLocalized (localized name of the country, non-persistent)
- officialNameEn (official English name)
- officialNameLocal (official name of the country in local language and local script)
- parentTerritoryUnCodeNumber (UN number of the territory in which the country is located)
- phonePrefix (international phone prefix for the country)
- shortNameEn (short English name)
- shortNameLocal (short name of the country in local language and local script)
- topLevelDomain (Internet top level domain of the country)
- unMember (whether or not the country is a member of the UNO)
- additional method: isUnMember()
- zoneFlag (whether or not the country has subdivisions)

CountryZone

The class is named: SJBR\StaticInfoTables\Domain\Model\CountryZone

The following properties have a getter and a setter:

- countryIsoCodeA2
- countryIsoCodeA3
- countryIsoCodeNumber
- deleted (whether or not the country zone was removed from the ISO standard)
- isoCode
- localName (name of the country zone in local language and local script)
- nameEn (English name of the country zone)
- nameLocalized (localized name of the country zone, non-persistent)

Currency

The class is named: SJBR\StaticInfoTables\Domain\Model\Currency

The following properties have a getter and a setter:

- decimalDigits (number of decimals to be shown when an amount is presented in this currency)
- decimalPoint (character to be shown in front of the decimals when an amount is presented in this currency)
- deleted (whether or not the currency was removed from the ISO standard)
- divisor (divisor used to obtain the subdivision of the currency)
- isoCodeA3
- isoCodeNumber
- nameEn (English name of the currency)
- nameLocalized (localized name of the currency, non-persistent)
- subdivisionNameEn (English name of the subdivision of the currency)
- subdivisionSymbolLeft (symbol to be shown to the left of an amount stated in units of the subdivision of the currency)
- subdivisionSymbolRight (symbol to be shown to the right of an amount stated in units of the subdivision of the currency)
- symbolLeft (symbol to be shown to the left of an amount stated in units of the currency)
- symbolRight (symbol to be shown to the right of an amount stated in units of the currency)
- thousandsPoint (Character to be used between every group of thousands of an amount stated in units of this currency)

Language

The class is named: SJBR\StaticInfoTables\Domain\Model\Language

The following properties have a getter and a setter:

- collatingLocale
- constructedLanguage (whether or not the language is a constructed language)
- additional method: isConstructedLanguage()
- countryIsoCodeA2 (country code as two digit string that identifies this language as a variant of the language identified by property isoCodeA2)
- deleted (whether or not the language was removed from the ISO standard)
- isoCodeA2
- localName (name of the language in the language itself)
- nameEn (English name of the language)
- nameLocalized (localized name of the language, non-persistent)
- typo3Code (deprecated)
- sacredLanguage (whether or not the language is a sacred language)
- additional method: isSacredLanguage()
- typo3Code (deprecated)

Domain model extension by language pack

Additional properties

Language packs will add properties and corresponding getters and setters to the domain model objects in the following way.

A language pack is identified by a two-letter ISO language code (e.g. de), possibly followed by an underscore and a two-letter ISO country code (e.g. de_AT).

Let XX stand for the camelization of the identifier of the language pack (e.g. De or DeAt), the following properties will be added to the domain model objects:

- SJBR\StaticInfoTables\Domain\Model\Territory: nameXX
- SJBR\StaticInfoTables\Domain\Model\Country: shortNameXX
- SJBR\StaticInfoTables\Domain\Model\CountryZone: nameXX
- SJBR\StaticInfoTables\Domain\Model\Currency: nameXX and subdivisionNameXX
- SJBR\StaticInfoTables\Domain\Model\Language: nameXX

Localization property nameLocalized

The property nameLocalized is available on each of the static entities of the domain model. The property is derived from the available naming properties and is non-persistent.

This method getNameLocalized() applies the localization rules as specified by the current TYPO3 configuration, using any available localized names as provided by the installed language packs.

Repositories and methods

Abstract repository

The following methods are available on the repository of all static entities.

localizedSort(\TYPO3\CMS\Extbase\Persistence\QueryResultInterface $entities)

Returns an array of all input entities ordered by localized name.

findAllOrderedBy($fieldName, $orderDirection = 'asc')

Returns all domain objects ordered by the given field name and the given order direction.

Territory

The repository class is named: SJBR\StaticInfoTables\Domain\Repository\Territory

The following methods are available:

findByCountry(\SJBR\StaticInfoTables\Domain\Model\Country $country)

Returns the territory in which the country is located.

findByTerritory(\SJBR\StaticInfoTables\Domain\Model\Territory $territory)

Returns the territory in which the territory is located.

findWithinTerritory(\SJBR\StaticInfoTables\Domain\Model\Territory $territory)

Returns all territories within a territory recursively.

Country

The repository class is named: SJBR\StaticInfoTables\Domain\Repository\Country

The following methods are available:

findByTerritory(\SJBR\StaticInfoTables\Domain\Model\Territory $territory)

Returns all countries located in the given territory

findByTerritoryOrderedByLocalizedName(\SJBR\StaticInfoTables\Domain\Model\Territory $territory)

Returns all countries located in the given territory ordered by localized name.

CountryZone

The repository class is named: SJBR\StaticInfoTables\Domain\Repository\CountryZone

The following methods are available:

findByCountry(\SJBR\StaticInfoTables\Domain\Model\Country $country)

Returns all country zones of the country.

findByCountryOrderedByLocalizedName(\SJBR\StaticInfoTables\Domain\Model\Country $country)

Returns all country zones of the country ordered by localized name.

Currency

The repository class is named: SJBR\StaticInfoTables\Domain\Repository\Currency

The following method is available:

findByCountry(\SJBR\StaticInfoTables\Domain\Model\Country $country)

Returns the currency in use in the country.

Language

The repository class is named: SJBR\StaticInfoTables\Domain\Repository\Language

The following methods are available:

findAllNonConstructedNonSacred()

Returns all language objects of non-contructed and non-sacred languages.

findOneByIsoCodes($languageIsoCodeA2, $countryIsoCodeA2 = '')

Returns the language object with the specified language and, optionally, country ISO codes.

Using the model

Adding a country select field to a form

If you want to insert a country select box into a form, you first have to add a Fluid select view helper to your Fluid template:

```
<f:form.select name="country" options="{countries}" optionLabelField="nameLocalized"
sortByOptionLabel="1" />
```

Using property “nameLocalized” as label will display the country names is the language of the current page.

In the script of the controller class, you then need to inject the Country repository:

```
/**
 * @var \SJBR\StaticInfoTables\Domain\Repository\CountryRepository
 */
 protected $countryRepository;

/**
 * Dependency injection of the Country Repository
 *
 * @param \SJBR\StaticInfoTables\Domain\Repository\CountryRepository $countryRepository
 * @return void
 */
 public function injectCountryRepository(\SJBR\StaticInfoTables\Domain\Repository\CountryRepository
$countryRepository) {
      $this->countryRepository = $countryRepository;
}
```

Finally, your action method needs to fill in the countries:

```
public function myNewAction() {
      $countries = $this->countryRepository->findAll();
      $this->view->assign('countries', $countries);
}
```

Using the select View Helper

Alternatively, the extension provides a viewHelper to insert a select field on a form.

Default usage:

```
{namespace s=SJBR\StaticInfoTables\ViewHelpers}
<s:form.select name="staticInfoTablesTestCountry" staticInfoTable="country" options="{}" />
<s:form.select name="staticInfoTablesTestLanguage" staticInfoTable="language" options="{}" />
<s:form.select name="staticInfoTablesTestTerritory" staticInfoTable="territory" options="{}" />
<s:form.select name="staticInfoTablesTestCurrency" staticInfoTable="currency" options="{}" />
<s:form.select name="staticInfoTablesTestCountryZones" staticInfoTable="countryZone" options="{}" />
```

Optional usage:

```
<s:form.select name="staticInfoTablesTestCountry" id="staticInfoTablesTestCountry"
staticInfoTable="country" options="{}" optionLabelField="isoCodeA2" />
<s:form.select name="staticInfoTablesTestCountry" id="staticInfoTablesTestCountry"
staticInfoTable="country" options="{}" optionLabelField="capitalCity" />
```

Subselect usage: (only CountryZones of Germany)

```
<s:form.select name="staticInfoTablesTestCountryZones" id="staticInfoTablesTestCountryZones"
staticInfoTable="countryZone" options="{}" staticInfoTableSubselect="{country: 54}" />
```

Further examples

You may find further examples in the source code of the test form included in the backend module of this extension.

Configuration of backend forms

Localization of entity selector

When configuring an entity selector, it is possible to render a localized selector using the following class in the TCA itemsProcFunc configuration property: SJBR\StaticInfoTables\Hook\Backend\Form\FormDataProvider\TcaSelectItemsProcessor.

For example, the following will render a localized countries selector:

```
$GLOBALS['TCA'][tablename]['columns'][fieldname]['config'] = array(
      'type' => 'select',
      'items' => array(
            array('', 0),
      ),
      'foreign_table' => 'static_countries',
      'foreign_table_where' => 'ORDER BY static_countries.cn_short_en',
      'itemsProcFunc' => 'SJBR\\StaticInfoTables\\Hook\\Backend\\Form\\FormDataProvider\\
TcaSelectItemsProcessor->translateCountriesSelector',
      'size' => 1,
      'minitems' => 0,
      'maxitems' => 1
);
```

Note: SJBR\StaticInfoTables\Hook\Backend\Form\FormDataProvider\TcaSelectItemsProcessor applies to TYPO3 CMS 7. For TYPO3 CMS 6.2, use SJBR\StaticInfoTables\Hook\Backend\Form\ElementRenderingHelper.

Alternate value fields for entity selector

When configuring an entity selector as a localized selector, it is also possible to use fields other than the uid for the selector values. This is done by adding the itemsProcFunc_config array and the indexField property.

For example, the following will render a localized languages selector. The value of each language will be the value of the language ISO Code A2. If the language has a country ISO Code A2, it is concatenated, the two codes being separated by an underscore (such as in PT_BR for Brazilian Protuguese).

```
$GLOBALS['TCA'][tablename]['columns'][fieldname]['config'] = array(
      'type' => 'select',
      'items' => array(
            array('', 0),
      ),
      'foreign_table' => 'static_languages',
      'allowNonIdValues' => TRUE,
      'foreign_table_where' => 'AND static_languages.lg_sacred = 0 ORDER BY
static_languages.lg_name_en',
      'itemsProcFunc' => 'SJBR\\StaticInfoTables\\Hook\\Backend\\Form\\FormDataProvider\\
TcaSelectItemsProcessor->translateLanguagesSelector',
      'itemsProcFunc_config' => array(
            'indexField' => 'lg_iso_2,lg_country_iso_2',
      ),
      'size' => 1,
      'minitems' => 0,
      'maxitems' => 1
);
```

Note: SJBR\StaticInfoTables\Hook\Backend\Form\FormDataProvider\TcaSelectItemsProcessor applies to TYPO3 CMS 7. For TYPO3 CMS 6.2, use SJBR\StaticInfoTables\Hook\Backend\Form\ElementRenderingHelper.

Localization of suggest wizard

It is also possible to provide a localized suggest wizard for the selector:

For example, the following will add a localized suggest wizard to a localized countries selector:

```
$GLOBALS['TCA'][tablename]['columns'][fieldname]['config'] = [
    'type' => 'group',
    'internal_type' => 'db',
    'allowed' => 'static_countries',
    'foreign_table' => 'static_countries',
    'foreign_table_where' => 'ORDER BY static_countries.cn_short_en',
    'suggestOptions' => [
        'default' => [
            'pidList' => '0'
        ]
    ],
    'fieldWizard' => [
        'recordsOverview' => [
            'disabled' => true
        ],
        'tableList' => [
            'disabled' => true
        ]
    ],
    'size' => 1,
    'minitems' => 0,
    'maxitems' => 1
];
```

With TYPO3 6 and 7 LTS, this was:

```
$GLOBALS['TCA'][tablename]['columns'][fieldname]['config'] = array(
      'type' => 'select',
      'items' => array(
            array('', 0),
      ),
      'foreign_table' => 'static_countries',
      'foreign_table_where' => 'ORDER BY static_countries.cn_short_en',
      'itemsProcFunc' => 'SJBR\\StaticInfoTables\\Hook\\Backend\\Form\\FormDataProvider\\
TcaSelectItemsProcessor->translateCountriesSelector',
      'size' => 1,
      'minitems' => 0,
      'maxitems' => 1,
      'wizards' => array(
            'suggest' => array(
                  'type' => 'suggest',
                  'default' => array(
                        'receiverClass' => 'SJBR\\StaticInfoTables\\Hook\\Backend\\Form\\Wizard\\
SuggestReceiver'
                  )
            )
      )
);
```

Note: SJBR\StaticInfoTables\Hook\Backend\Form\FormDataProvider\TcaSelectItemsProcessor and SJBR\StaticInfoTables\Hook\Backend\Form\Wizard\SuggestReceiver apply to TYPO3 CMS 7.

Pre-Extbase API

API class

Class SJBR\StaticInfoTables\PiBaseApi (previously known as tx_staticinfotables_pi1) may be used in any TYPO3 frontend plugin or backend module.

When used in the frontend, some default values may be configured using the TS template constant editor. See the Configuration section.

Class initialization

The following example statements will instantiate and initialize the class:

```
$this->staticInfo = \TYPO3\CMS\Core\Utility\GeneralUtility::makeInstance('SJBR\\StaticInfoTables\\
PiBaseApi');
if ($this->staticInfo->needsInit()) {
      $this->staticInfo->init();
}
```

Method init() will initialize:

- the language of display: using the TypoScript template config.language property;
- the default currency used to format amounts: using the configured currencyCode;
- defaults for selected values in drop-down lists: using the configured countryCode, countryZoneCode, currencyCode and languageCode.

Getting a localized name

Each of the following example statements will return the name of some entity given its code. The name will be in the language specified by the TypoScript config.language property.

```
$this->staticInfo->getStaticInfoName('COUNTRIES', $countryCode);
$this->staticInfo->getStaticInfoName('SUBDIVISIONS', $zoneCode, $countryCode);
$this->staticInfo->getStaticInfoName('CURRENCIES', $currencyCode);
$this->staticInfo->getStaticInfoName('LANGUAGES', $languageCode, '', '', $self);
```

In the case of LANGUAGES, if $self is set to 1, the name of the language will be returned in the language itself.

See the source code for a description of the parameters.

Creating a localized drop-down selector

Each of the following example statements will return a string of <select> and <option> HTML tags to display a drop-down selector of countries, country subdivisions, currencies or languages. The drop-down selector will be produced in the language specified by the TS config.language property.

```
$this->staticInfo->buildStaticInfoSelector('COUNTRIES', $fieldName, $CSSClassName, $selectedCountryCode,
'', $submitFlag, $id, $title, $where, $lang, $local);
$this->staticInfo->buildStaticInfoSelector('SUBDIVISIONS',  $fieldName, $CSSClassName,
$selectedZoneCode, $countryCode, $submitFlag, $id, $title);
$this->staticInfo->buildStaticInfoSelector('CURRENCIES', $fieldName, $CSSClassName,
$selectedCurrencyCode, '', $submitFlag, $id, $title);
$this->staticInfo->buildStaticInfoSelector('LANGUAGES', $fieldName, $CSSClassName,
$selectedLanguageCode, '', $submitFlag, $id, $title);
```

See the source code for a description of these and other parameters.

Note that the entries of the drop-down selector are sorted using the current locale. If the front end page is using utf-8 encoding (config.metaCharset = utf-8), then the locale should be fully qualified; for example: config.locale_all = fr_CA.UTF-8.

Formating an amount

The following statement will return the amount $amount formated for display in the default currency or in the currency specified in the previously called loadCurrencyInfo method.

```
$formatedAmount = $this->staticInfo->formatAmount($amount);
```

Loading an alternate currency

The currency to be used to format amounts may be overridden using method loadCurrencyInfo

```
$this->staticInfo->loadCurrencyInfo($currencyCode);
```

where $currencyCode is the ISO alpha-3 code (cu_iso_3) of some currency.

The loaded currency will be effective until the same method is called again with a different currency code.

Method init() loads the default currency.

Formating an address

The following statement will return an address formated for display according to the address format specified for the country in the static_countries table (cn_address_format).

```
$formatedAddress = $this->staticInfo->formatAddress($delim, $streetAddress, $city, $zip,
$subdivisionCode, $countryCode);
```

where $delim will often be chr(10), $subdivisionCode is country subdivision code, and $countryCode is a ISO alpha-3 country code.

Getting country info

This method is deprecated and will be removed in version 6.2. Use the methods provided by \SJBR\StaticInfoTables\Domain\Repository\CountryRepository.

You can get the info for different countries by applying the parameters you have obtained from a FE edit field.

```
$countryArray = tx_staticinfotables_div::fetchCountries($input['country'], $input['countryISO2'],
$input['countryISO3'], $input['countryISONr']);
```

where the first parameter may be the text of the country in English or the local language and only parts of it. The fitting languages will be guessed by the input text.

Creating/updating a language pack

Creating a new language pack

A language pack may be created for any language supported by TYPO3.

These are the steps for creating a new language pack:

1. Use the backend module action “Create or update a language pack extension for the Static Info Tables” to initialize a new language pack; the new language pack is immediately installed upon its creation: fields are added to the tables and properties are added to the domain model;
2. Use the backend forms, at the root level of the TYPO3 installation, to translate the labels in the language-specific fields of each of the records of the Static Info Tables;
3. When updating the localized labels is complete, use the same action of the backend module to update the language pack with the localized data;
4. You may test the result of localization by setting the language of TYPO3 backend to the language of the language pack and by using the backend module action “Display the test form”.
5. After testing, the language pack is ready to be published on TYPO3 Extension Repository (TER).

Updating an existing language pack

These are the steps for updating an existing language pack:

1. Use the Extension Manager to install the existing language pack; upon installation, fields are added to the tables; if the language pack was created with version 6+ of Static Info Tables, properties are added to the domain model;
2. Use the backend forms, at the root level of the TYPO3 installation, to update the labels in the language-specific fields of each of the records of the Static Info Tables;
3. When updating the localized labels is complete, use the backend module action “Create or update a language pack extension for the Static Info Tables” to update the language pack with the updated localized data;
4. You may test the result of localization by setting the language of TYPO3 backend to the language of the language pack and by using the backend module action “Display the test form”.
5. After testing, the language pack is ready to be published on TYPO3 Extension Repository (TER).

Installation

Installing the extension

Install the extension using the Extension Manager.

The update script should be run every time the extension is installed or upgraded.

The extension has a configuration option that allows to enable the Static Info Tables Manager backend module. The backend module allows to create language packs and to test the installation.

Once the extension is installed any available language pack may be installed. The update script of the language pack should be run every time the language pack is installed or upgraded.

After installing the extension and any language packs, the test form provided by the backend module may be used to verify that the localization is working as expected.

Upgrading to version 12.4.1+

Value and use of TS constant onChangeAttribute was changed

In order to be compatible with Content Security Policy, no onchange attribute is added anymore on generated html select elements, in particular on the country selector. The value of the constant onChangeAttribute is rather inserted into the body of an event handler that is inserted on the page as inline JavaScript with appropriate nonce attribute. Therefore the default value of the constant was changed from “this.form.submit();” to “event.target.form.submit();

Upgrading to version 6.0+ from older versions

Static template “Static Info Tables” was moved

Static template “Static Info Tables” was moved and therefore needs to be re-included in any TS template in which it was previously included.

Language packs need to be re-created

Language packs that were not created with the module provided by version 6.0 will not extend the domain model. Therefore some features will not be available to Extbase/Fluid extensions.

Until a language pack gets updated on TER, administrators may work around this issue by:

1. installing the existing language pack;
2. updating the langugage pack using the Static Info Tables backend module.

Class tx_staticinfotables_div is removed

Class tx_staticinfotables_div is removed in version 6.3.

The scripts that use this class should be modified to use \SJBR\StaticInfoTables\Utility\LocalizationUtility or \SJBR\StaticInfoTables\Domain\Repository\CountryRepository depending on the method being invoked.

The itemsProcFunc tx_staticinfotables_div->selectItemsTCA that could be used in TCA configurations to generate a hotlist of items will not add any items anymore. Therefore TCA configurations using this itemsProcFunc should be modified. A suggest wizard may be used as a replacement for the hotlist feature that was removed. See section “Configuration of backend forms”.

Configuration

Extension configuration

The extension has a configuration option that allows to enable the Static Info Tables Manager backend module. The module allows to create language packs.

TypoScript reference

Use the Constant Editor template tool to set these properties used by front end API:

Property:	Data type:	Description:	Default:
currencyCode	string	The ISO alpha-3 code of the currency to be used to format amounts.	EUR
countryCode	string	ISO alpha-3 code of the default selected country in drop-down selector.	
countriesAllowed	string	ISO alpha-3 code of the countries which are allowed in the drop-down selector. The order of the countries will remain in the output. In the following example, the output in the select box will be 'Germany | Austria | Swizzerland'. **Example:** `countriesAllowed = DEU,AUT,CHE`	
countryZoneCode	string	Code of the default selected country zone in drop-down selector.	
languageCode	string	ISO alpha-2 code of the default selected language in drop-down selector. May also be of the form LG_CN where LG is an ISO alpha-2 language code and CN an ISO alpha-2 country code.	
onChangeAttribute	string	Script to be executed by the onChange event handler of html select elements generated by the extension.	event.target.form.submit();

Address formats

Nine (9) address formats are defined in TS setup of static template “Static Info Tables” as follows:

```
addressFormat {
        ## See www.upu.int
        ## Semi-colon (;)-separated address lines
        ## Examples of address format 1: Austria, Denmark, France, Germany, Russia
           1 = %street;%zip  %city;%countryName
        ## Examples of address format 2: India
           2 = %street;%city %zip;%countryName
        ## Examples of address format 3: Australia, USA
           3 = %street;%city %countrySubdivisionCode  %zip;%countryName
        ## Example of address format 4: Canada
           4 = %street;%city (%countrySubdivisionName) %zip;%countryName
        ## Example of address format 5: Great Britain
           5 = %street;%city;%zip%countryName
        ## Example of address format 6: Mexico
           6 = %street;%zip %city, %countrySubdivisionCode;%countryName
        ## Example of address format 7: Italy
           7 = %street;%zip %city %countrySubdivisionCode;%countryName
        ## Example of address format 8: Spain
           8 = %street;%zip %city (%countrySubdivisionName);%countryName
        ## Example of address format 9: Brazil
           9 = %street;%city - %countrySubdivisionCode;%zip;%countryName
}
```