TCAobjects

Fabrizio Branca

mail@fabrizio-branca.de

I started implementing my tcaobjects extension…

… and when almost finished I found this:

[...] The general idea with our mapper (called tcaObj) is that the TCA already contains the info we need to read and write data from the database (table name, field names, field types, validation requirements, etc). The backend proves this with the use of TCEMain and TCEForms. [...]

… and a few minutes later:

[...] Jeff, this is awesome news! I was going to work on exactly this after my thesis is done (still a long way to go), but the similarities are really striking (btw. even my class is called TCAobject ;)). [...]

What‘s this all about?
Some buzzwords…

TCA,IRRE, HTML_QuickForm, DDD, Ruby on Rails, Database Relations, TYPO3, Active Record, OOP, PHP5, Smarty, RAD, CakePHP, ORM, MVC, CRUD, Forms, Persistence, Magic Methods, Interfaces, Kickstarter, Autoloader, Chaining, Lazy loading

The Idea behind the extension…

- While programming most time is spent with doing trivial things like
  - …writing class definitions (including accessor classes, collection classes,…)
  - …writing getter/setter methods
  - …writing language labels
  - …transfering variables into templates
  - …adapting properties to table definitions and TCA
  - …

The Idea behind the extension…

- Most of these tasks
  - contain redundant definitions
  - are boring and time consuming
  - are error-prone
  - **can be avoided!**
- The „M“ in MVC

Other Frameworks with simalar approaches

- Ruby on Rails
- CakePHP
- Propel

New possibilities with PHP5

- Magic methods
  - _get(), _set(), _call()
- Interfaces:
  - Iterators, „ArrayAccess“
  - Define own interface
- Abstract classes

Why not define
everything at **one place**?

Fabrizio Branca – mail@fabrizio-branca.de

Active Record Pattern

- Martin Fowler in „ Patterns of Enterprise Application Architecture“:

- „An object that
  - **wraps a row** in a database table or view,
  - encapsulates the **data access**
  - and adds **domain logic** on that data“

Active Record Pattern

- A class corresponds to a table definition
- An instance corresponds to a record
- Domain logic is added to the class
- The class does CRUD operations in the background
- Used for object relational mapping (ORM)
- Ruby on Rails: „ActiveRecord“

Using the information in the TCA

- Field types
- Database relations („old style“ and IRRE)
- Evaluation

class „tx_tcaobjects_object“

- abstract class
- main parent class for all objects
- class name == table name
  - or: `$this->_table = ‘otherTableName‘;`
- implements interfaces
  - ArrayAccess
  - IteratorAggregate

tx_tcaobjects_object: **Basic functions**

- Accessing properties
  - Dynamic functions
    - `echo $this->get_message();`
      `$this->set_message('Hello World');`
  - ArrayAccess
    - `echo $this['message'];`
      `$this['message'] = 'Hello World';`
  - IteratorAggregate
    - ```
      foreach ($obj as $key => $property) {
        echo $key .': '. $property;
      }
      ```
      ```

tx_tcaobjects_object: **Loading and storing data**

- Via constructor
  - $myObj = new tx_myext_myclass(42);
- loadSelf() / storeSelf() methods
  - ```
    $myObj = new tx_myext_myclass();
    $myObj['foo'] = 'bar';
    $myObj->storeSelf();
    ```
    ```

tx_tcaobjects_object: **Aliases**

- ```
  class myClass extends fe_users {
      protected $_aliasMap = array(
            'firstname' => ' myClass_firstname',
            'lastname' => ' myClass_lastname');
  […]
  }
  ```

- ```
  $myObj = new myClass();
  $myObj['firstname'] = 'Fabrizio';
  echo $myObj['firstname']; // Outputs: Fabrizio
  echo $myObj['myClass_firstname']; // Outputs: Fabrizio
  ```

tx_tcaobjects_object
Dynamic properties

- by overriding the __get method:
-
```
protected function __get($calledProperty){

  switch ($calledProperty) {

    case 'fullName' :
      return $this['firstname'].' '.$this['lastname'];
    break;

    default: return parent::__get($calledProperty);

  }
}
```

tx_tcaobjects_object
Override TCA settings

- …locally (on a „per class“ basis)

```
protected $_properties = array(
  'image' => array (
    'config' => array (
      'maxitems' => 1,
      'size' => 1
    )
  ),
  'title' => array (
    'label' => 'LLL:EXT:myext/locallang_db.xml:myNewLabel‘
  )
);
```

tx_tcaobjects_object
Set default values

- ```
  class myext_blogpost extends tx_tcaobjects_object{

      protected $_table = 'tt_news';

      protected $_values = array('type' => 3);

  }
  ```
  ```

tx_tcaobjects_object
Modifiers

- Modifiers are appended with an „_“
- Only valid for special configurations
- Can be added in extending classes
- Modifier values will be „lazy loaded“

tx_tcaobjects_object
Modifiers

- Current available modifiers for relations
  - obj: Returns related object instead of uid to it
  - objColl: Creates a collection with related objects

tx_tcaobjects_object
Modifiers

- Current available modifiers (other)
  - path: returns the complete path in case of files
  - explode: returns an array in case of csl
  - rte: renders content as rte content
  - sL: returns the language label

tx_tcaobjects_object
Chaining

- The name of the artist of a users first album:

```
$userObj['albums_objColl'][0]['artist_obj']['name'];
```

class „tx_tcaobjects_objectCollection“

- „group/list/set“ of objects
- implements some interfaces
  - ArrayAccess
  - tx_tcaobjects_iPageable
  - IteratorAggregate (from tx_pttools_objectCollection)
  - Countable (from tx_pttools_objectCollection)

class „tx_tcaobjects_objectCollection“

- ```
  foreach ($myObjColl as $myObj) {
    /* @var $myObj tx_myExt_myClass */
    echo $myObj['someProperty'];
  }
  ```
  ```

class „tx_tcaobjects_objectAccessor“

- All methods for database operations (select, insert, update, delete) are bundled here
- Put your sql statements in accessor classes
- only static methods

Other Features

- Autoloader

```
$GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['tcaobjects']
['autoLoadingPath'][str_replace('_','',$_EXTKEY)] = 'EXT:'.
$_EXTKEY.'/res/';
```

- Assertions (tx_tcaobjects_assert)

```
tx_tcaobjects_assert::notEmpty($this->piVars['album_uid']);
```

  - throws an exception if assertion fails
- tx_tcaobjects_fe_users
  - is able to wrap the current logged in user into an object

Forms with HTML_QuickForm

- HTML_QuickForm is part of PEAR
  - Classes must be in your include path (e.g. use EXT:pear)
- Allows to create forms easily
- class tx_tcaobjects_quickform
- Interface for renderers:
  class.tx_tcaobjects_iQuickformRenderer.php
  - tx_tcaobjects_qfDefaultRenderer
  - tx_tcaobjects_qfSmartyRenderer

Forms with HTML_QuickForm

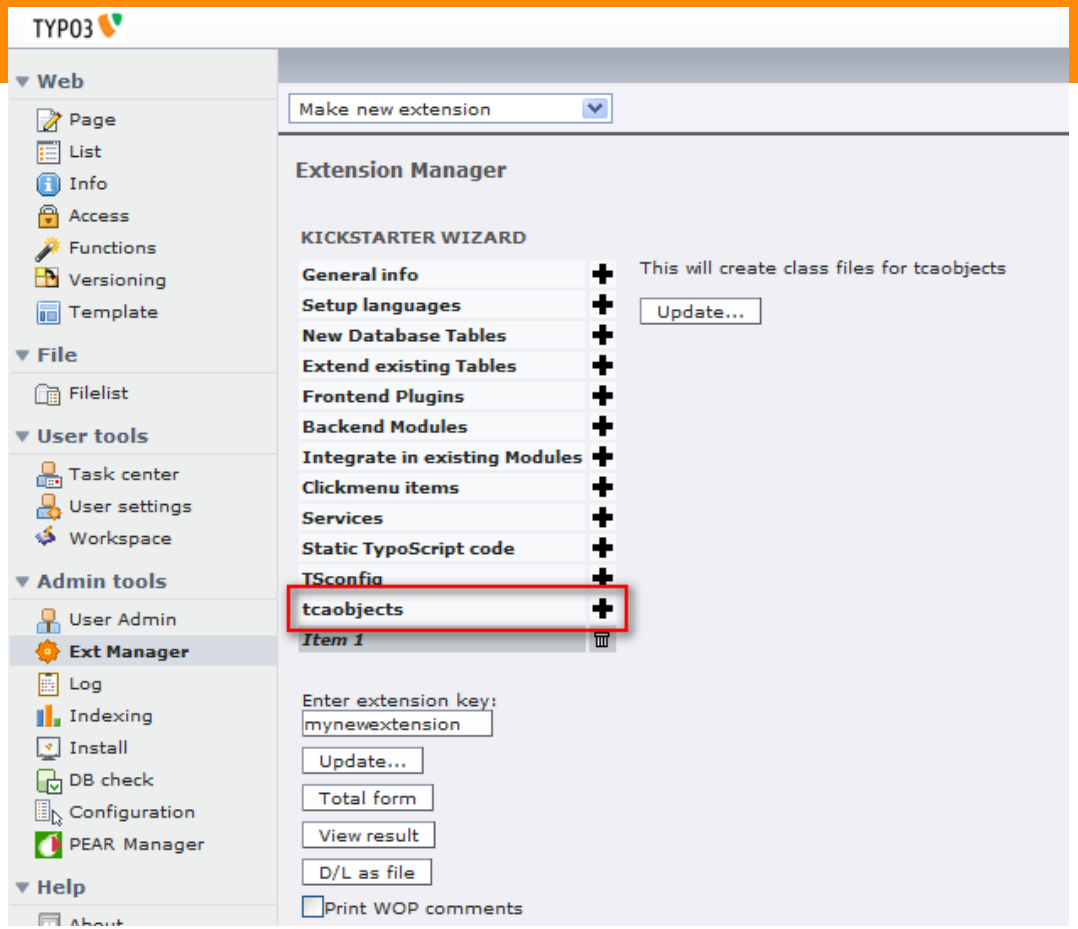
- Define forms
  - with PHP
  - with a string (similar to the „showitem“ string)
  - by Typoscript
- Use filters and rules (defined e.g. in TCA)
- Create own rules and filters
  - e.g. check full age

Templating with Smarty

- The „V“ in MVC
- Use TYPO3 functions to render content
  - New features of the smarty extension
    - {link}
    - {image}
  - New features that come with tcaobjects
    - {ll}

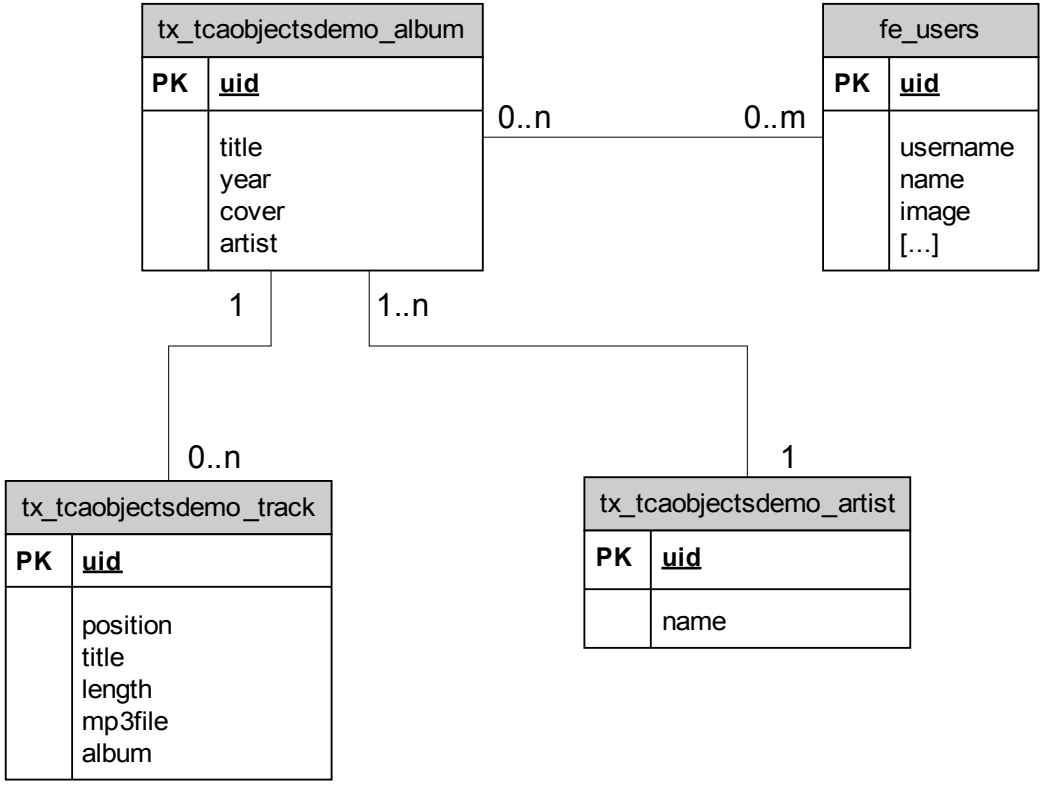
Developing an extension

- tcaobjects extends the kickstarter
- classes will be generated automatically for all new and extend tables
  - objects
  - collections
  - accessors



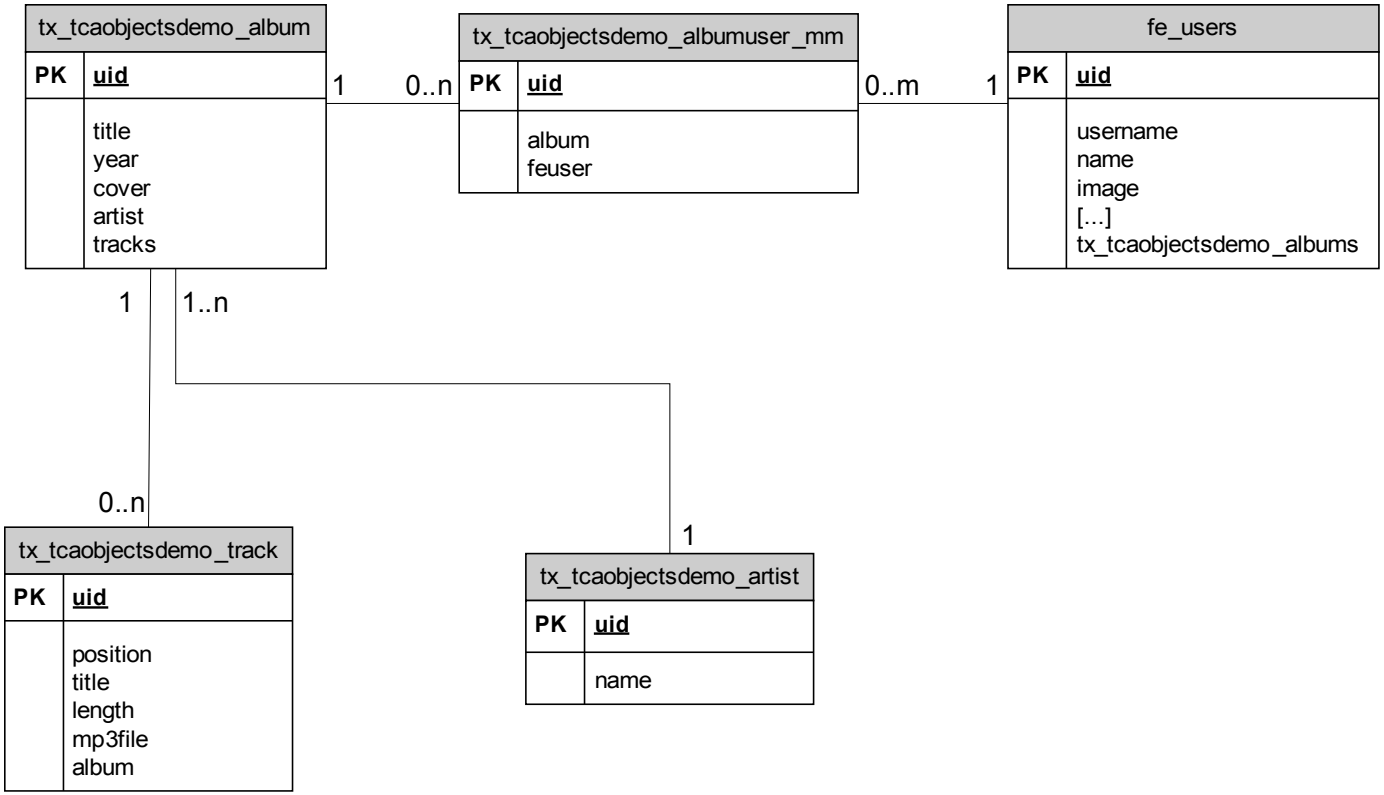
Fabrizio Branca – mail@fabrizio-branca.de

Demo Extension



Fabrizio Branca – mail@fabrizio-branca.de

Demo Extension



Fabrizio Branca – mail@fabrizio-branca.de

Demo

What now?

- Very experimental. Do not use in production environment now!
- Developing a community project based on tcaobjects. So expect this extension to maturate soon… :)

Resources

- Martin Fowler
  „Patterns of Enterprise Application Architecture“
- ActiveRecord Tutorial (Ruby on Rails)
  http://www.railsenvy.com/2007/8/8/activerecord-tutorial
- HTML_QuickForm
  http://pear.php.net/package/HTML_QuickForm
- Smarty
  http://smarty.php.net

Resources

- TER:
  - tcaobjects
  - tcaobjects_demo
- Manual will follow soon
- These slides will be available at http://www.fabrizio-branca.de

Questions?

mail@fabrizio-branca.de