TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Technická dokumentace PHP frameworku

Vytvořeného pro bakalářskou práci – Matrika pro mateřské školy

Liberec 2013 **Pavel Bucháček**

Obsah

1 Struktura frameworku

1.1 Návrhový vzor Registry

Framework využívá návrhového vzoru Registry. Ten je realizován třídou, která implementuje návrhový vzor Singleton. V systému tedy existuje vždy pouze jedna instance tohoto objektu. Samotná třída slouží pro distribuci jednotlivých modulů frameworku a nastavení napříč aplikací. Obsahuje tedy kolekci s referencemi na vytvořené objekty frameworku a pole s nastavením aplikace. Obsahuje taktéž metody, které nejsou vázány na konkrétní část frameworku.

1.2 Návrhový vzor FCP

Další návrhový vzor, který framework využívá je FCP – všechny požadavky tedy vyřizuje jeden bootstrap soubor.

1.3 Návrhový vzor MVC

Framework využívá architektonického modelu MVC, který důkladně rozděluje aplikaci na datovou vrstvu, prezentační a vrstvu, která se stará o vyřizování požadavků mezi předchozími vrstvami.

1.4 Konfigurační soubory config.php a .htaccess

V souboru config.php jsou uloženy všechny potřebná nastavení aplikace, například přihlašovací údaje do databáze. Vzhledem k tomu, že framework využívá vlastní modul pro směrování požadavků, slouží soubor .htaccess pouze pro přesměrování všech požadavků na bootstrap soubor. Původní hodnota adresy je při tom skriptu předávána ve formě GET proměnné s názvem url.

1.5 Bootstrap soubor index.php

V bootstrap souboru dochází k definici konstant, pomocí direktivy `set_include_path` se nastaví cesty ke všem zdrojovým kódům aplikace (pro funkci `__ autoload`). Poté dojde k načtení konfiguračního souboru, vložení všech dostupných modulů a nastavení do registru a vytvoření spojení s databází. Před předání řízení routeru může ještě dojít k automatickému generování struktury databáze a dat.

1.6 Adresářová struktura aplikace

- controllers – Zde se nachází všechny kontrolery aplikace.
- database_utils – Sada tříd starajících se o generování databáze.
- locale – Databázové soubory s překlady pro různé jazykové mutace.
- model – Zde se nachází všechny modely aplikace.
- registry – Adresář s jednotlivými částmi frameworku.

    - objects
        - database
        - forms
- services – Servisní třídy modelů aplikace.
    - i18n – Utility pro internacionalizaci aplikace s využitím Gettext.
- tests – Testovací skripty.
- users_data – Libovolná data, které aplikace potřebuje ke svému provozu.
- views – Složka se všemi šablonami aplikace.
    - default – Výchozí vzhled aplikace, pro další vzhled je nutné vytvořit další adresář se shodnou strukturou.
        - css
        - img
        - templates
    - defaultForms – Výchozí šablony pro formuláře.
    - js – JavaScriptové soubory.
    - plugins – JavaScriptové doplňky třetích stran (např.: jQuery).

2 Třída Registry

Jmenný prostor:	`Bs\Registry`
`buildSaltedHash($passwordHash, $salt)`	Sestaví salted hash pro vstupní textový řetězec a sůl (získána pomocí funkce getUniqueToken())
`customUrlDecode($input)`	Dekóduje řetězec z url adresy.
`customUrlEncode($input)`	Zakóduje řetězec pro korektní vložení do url adresy.
`debugBacktrace($debug_backtrace)`	Funkce pro výpis debugovací hlášky.
`findIndexInMultidimensionArray($search, $array, $indent = 0)`	Ekvivalent funkce in_array pro pole s více dimenzemi.
`getObject($index)`	Získá uložený objekt.
`getSetting($index)`	Získá uložené nastavení.
`getUniqueToken()`	Získá unikátní textový řezězec.
`insertObject($objectName, $objectIndex)`	Vloží objekt pod daným indexem.
`insertSetting($index, $value)`	Vloží nastavení pod daným indexem.
`removeGraphicAccent($str, $separator = "-")`	Z předaného řetězce odstraní veškerou diakritiku a mezery nahradí předaným oddělovačem.
`singleton()`	Vrací vytvořenou instanci registru. První instance se vytváří v bootstrap souboru a je potřeba ji distribuovat pomocí konstruktoru do každé třídy v aplikaci.

3 Práce s databází

Aby framework nebyl závislý na konkrétním typu databáze, existuje v systému abstraktní třída `Database`, od které dědí všechny konkrétní implementace databází. V současnosti je implementována databáze MySQL, která obsahuje následující metody:

Jmenný prostor:	`Bs\Registry\Objects\Database`
`beginTransaction()`	Zahájí databázovou transakci.
`buildSelect($select, $from, $where = "", $groupBy = "", $groupByAsc = true, $having = "", $orderBy = "", $limitOffset = "", $limitRowCount = "", $join = "")`	Sestaví příkaz pro selektování dat z databáze. Přijímá parametry: • select: pole s názvy atributů • from: název tabulky • where: podmínka, je nutné použít instanci třídy Bs\Registry\Objects\Database\WhereCondition • groupBy – název atributu • groubByAsc – boolean hodnota • having – podmínka, je nutné použít instanci třídy Bs\Registry\Objects\Database\WhereCondition • orderBy – je nutné použít funkci pasteOrderBy • limitOffset a limitRowCount – čísla pro offset a počet řádků • join – spojení více tabulke, je nutné použít funkci pasteJoin
`closeAllConnection()`	Zavře všechna vytvořená spojení s databázemi.
`closeConnection($index)`	Zavře konkrétní spojení s databází.
`commit()`	Potvrdí otevřenou transakci.
`delete($table, $where)`	Smaže záznamy z tabulkyvyhovující podmínce. Parametry: • tabel – název tabulky • where – podmínka, instance třídy Bs\Registry\Objects\Database\WhereCondition
`getLastInsertId()`	Vrací id posledního vloženého záznamu.
`insert($table, $values)`	Vloží data do tabulky, vrací hodnotu id nového záznamu. Parametry: • table – název tabulky • values – pole s názvy atributů a jejich hodnotami s definovanou strukturou: values["název atributu"]="hodnota"

`newConnection($db_host, $db_name, $db_user, $db_pass, $db_charset)`	Vytvoří nové spojení s databází, které následně uloží do kolekce všech spojení. Vrací jeho index v poli spojení.
`orderQuery($sql, $retResult = false)`	Provede předaný SQL příkaz.
`orderSelectQuery($index)`	Provede příkaz vytvořený pomocí funkce buildSelect()
`pasteAs($name)`	Vloží SQL příkaz AS do select příkazu.
`pasteAvg($columnName)`	Vloží SQL příkaz AVG do select příkazu.
`pasteBetween($value1, $value2, $notBetween = false)`	Vloží SQL příkaz BETWEEN do select příkazu.
`pasteConcat($bits)`	Vloží SQL příkaz CONCAT pro spojení více řetězců do select příkazu.
`pasteCount($columnName)`	Vloží SQL příkaz COUNT do select příkazu.
`pasteCurDate()`	Vloží SQL příkaz pro získání aktuálního data.
`pasteCurTime()`	Vloží SQL příkaz pro získání aktuálního času.
`pasteDateAdd($date, $timeInterval, $numberOfIntervals)`	Vloží SQL příkaz pro přičtení časového úseku k datu.
`pasteDateDiff($date1, $date2)`	Vloží SQL příkaz pro rozdíl mezi dvěma daty..
`pasteDateFormat($date, $format)`	Vloží SQL příkaz pro zformátování data.
`pasteDateSub($date, $timeInterval, $numberOfIntervals)`	Vloží SQL příkaz pro odečtení dvou dat.
`pasteDistinct($attributeName)`	Vloží SQL příkaz DISTINCT.
`pasteFloor($columnName)`	Vloží SQL příkaz pro zaokrouhlení čísla.
`pasteGroupConcat($attribute, $orderBy = "", $orderByAsc = true, $separator = ", ")`	Vloží SQL příkaz GROUP CONCAT.
`pasteIn($columnName, $values, $notIn = false)`	Vloží SQL příkaz IN.
`pasteIsNull($columnName, $notNull = false)`	Vloží SQL podmínku na porovnání s null hodnotou.
`pasteJoin($table, $joinOn, $leftJoin = true, $outerJoin = false)`	Pro spojení více tabulek.
`pasteLike($pattern, $notLike = false)`	Pro vložení podmínky LIKE.
`pasteMax($columnName)`	Vloží SQL příkaz pro získání maximální hodnoty.
`pasteMid($columnName, $start, $length = -1)`	Vloží SQL příkaz pro získání průměrné hodnoty.

pasteMin($columnName)	Vloží SQL příkaz pro získání minimální hodnoty.
pasteNow()	Vloží SQL příkaz pro získání aktuálního času.
pasteOrderBy($orderByArray)	Vloží SQL příkaz pro řazení výsledků příkazů select.
pasteRand()	Vloží SQL příkaz pro získání náhodného čísla.
pasteRound($columnName, $decimals = 0)	Vloží SQL příkaz pro zaokrouhlení.
pasteStrLength($columnName)	Vloží SQL příkaz pro získání délky řetězce.
pasteSubSelect($index)	Do příkazu select vloží vnořený příkaz, který je uložen v kolekci dotazů pod předaným indexem.
pasteSubstring($attribute, $start, $length)	Vloží SQL příkaz pro získání podřetězce.
pasteSum($columnName)	Vloží SQL příkaz pro součet daného atributu.
pasteTimeStampDiff($unit, $lowerDate, $higherDate)	Vloží SQL příkaz pro rozdílu mezi dvěma daty.
pasteWeekDay($attribute, $indexOffset = +1)	Vrací index dne v týdnu pro předané datum.
rollback()	Zruší otevřenou transakci.
unionQueries($indexes, $orderBy = "")	Příkaz pro sjednocení více dotazů.
update($tableName, $set, $where = "")	Příkaz pro aktualizaci hodnot tabulky. Přijímá parametry: • table – název tabulky • values – pole s názvy atributů a jejich hodnotami s definovanou strukturou: values["název atributu"]="hodnota"

Ukázka vykonání příkazu select:

```
$id = $this->db->getResults(
      $this->db->orderSelectQuery(
         $this->db->buildSelect(
            Array("id"),
            history,
            new \Bs\Registry\Objects\Database\WhereCondition(
              "children_id = ?", $childrenId)
          )
         )
      );
```

4 Práce s formuláři

4.1 Třída Forms

V registru je uložena instance třídy Forms, která obsahuje všechny vytvořené formuláře a obsahuje také metody pro vytvoření nového formuláře.

Jmenný prostor:	Bs\Registry\Objects\Forms
`newForm($name, $id, $method = "post", $action = "", $enctype = "multipart/form-data")`	Metoda pro vytvoření nového formuláře. Přijímá parametry: • name – jméno formuláře • id – identifikátor formuláře • method – metoda odeslání dat • action – skript, který se provede po odeslání formuláře • enctype – typ odeslaných dat Metoda vrací instanci takto vytvořeného formuláře.
`__get()`	Třída implementuje funkci __get, takže po zavolání jména formuláře jako datové položky třídy dojde k vrácení instance formuláře.
`setActiveForm($name)`	Nastaví aktivní formulář.

Příklad vytvoření formuláře a získání jeho instance:

```
$forms = $this->registry->getObject("forms");
$forms->newForm("childrenForm", "childrenForm");
$childrenForm = $forms->childrenForm;
```

4.2 Třída Form

Třída, která reprezentuje konkrétní formulář a obsahuje všechny jeho prvky.

Jmenný prostor:	Bs\ Bs\Registry\Objects\Forms
`addButton($name, $value)`	Přidá tlačítko do formuláře.
`addCheckboxInput($name, $label, $values)`	Přidá checkbox do formuláře.
`addFileButton($name, $label)`	Přidá tlačítko pro nahrání souboru do formuláře.
`addGroup($name, $legend)`	Přidá skupinu, do které je možné seskupit prvky formuláře.

addHiddenInput($name, $value)	Přidá skrytí textové pole do formuláře.
addOwnErrorMessage($message)	Pokud dochází k externí validaci formuláře, je možné chybové hlášky vložit pomocí této funkce. Práce s nimi je poté shodná jako s nativními hláškami.
addPasswordInput($name, $label)	Přidá vstupní pole pro zadání hesla do formuláře.
addRadioInput($name, $label, $values)	Přidá prvek radio input do formuláře.
addSelect($name, $label, $values)	Přidá prvek select do formuláře.
addSubmitButton($name, $value)	Přidá tlačítko pro odeslání formuláře.
addTextInput($name, $label)	Přidá textové pole do formuláře.
addTextNoteAfterElement($text, $nameOfElement)	Vloží libovolnou textovou poznámku, která bude umístěna za prvek daného jména.
addTextNoteBeforeElement($text, $nameOfElement)	Vloží libovolnou textovou poznámku, která bude umístěna před prvek daného jména.
addTextarea($name, $label)	Přidá prvek textarea do formuláře.
getCssClass()	Vrací css třídu formuláře.
getElement($name)	Vrací prvek daného názvu.
getElements()	Vrací všechny prvky formuláře.
getId()	Vrací id formuláře.
getName()	Vrací jméno formuláře.
getSubmited()	Vrací boolean hodnotu v závislosti na tom, jestli došlo k odeslání formuláře.
getValidateMessages()	Vrací kolekci chybových zpráv o validaci.
setActiveGroup($name)	Nastaví aktivní skupinu prvků. Nové prvky se poté přidávají do takto nastavené skupiny.
setProtected($protected)	Nastaví ochranu formuláře pomocí htmlspecialchars.
setTemplate($template)	Nastaví cestu k šabloně formuláře.
setTemplateBlocks($templateBlocks)	Nastaví proměnné šabloně.
setTemplateTags($templateTags)	Nastaví proměnné šabloně.
submit()	Pokud formulář splňuje všechny validační pravidla, tak funkce vrací kolekci hodnot. V opačném případě vrací hodnotu null, chybové hlášky lze poté získat funkcí `getValidateMessages()`.

```
$forms = $this->registry->getObject("forms");
$forms->newForm("childrenForm", "childrenForm");
$childrenForm = $forms->childrenForm;
$childrenForm->setCssClass("form-horizontal");
$childrenForm->setLegend("Popis formuláře");
$childrenForm->addTextInput("lastName", "Příjmení")
   ->setValue("Výchozí hodnota")
   ->addValidationRule(\Bs\Registry\Objects\Forms\Form::VALIDATION_FILLED,
      "Musíte vyplnit příjmení.");
$childrenForm->addSubmitButton("submit", "Odeslat");

if ($childrenForm->getSubmited()) {
   $formData = $childrenForm->submit();
   if (is_null($formData)) {
      var_dump($childrenForm->getValidateMessages());
   } else {
      var_dump($childrenForm->submit());
   }
}
```

4.3 Validační konstanty

Každému prvku je možné pomocí funkce `addValidationRule` přidat validační pravidlo.

`addValidationRule($constant, $errorMessage, $value = "")`	• **constant – validační konstanta definovaná ve třídě Form** • **errorMessage – chybová zpráva, která se vypíše uživateli, pokud prvek není validní** • **value – volitelný parametr, který se předává některým typům validace**
Konstanty třídy **`Form:`**	
`VALIDATION_EMAIL`	Prvek je validní, pokud je předána korektní emailová adresa.
`VALIDATION_EMPTY`	Prvek je validní, pokud je prázdný.
`VALIDATION_EQUAL_TO`	Prvek je validní, pokud je jeho hodnota rovna předanému argumentu.
`VALIDATION_FILLED`	Prvek je validní, pokud je vyplněn.
`VALIDATION_FLOAT`	Prvek je validní, pokud je jeho hodnota desetinné číslo.
`VALIDATION_INTEGER`	Prvek je validní, pokud je jeho hodnota celé číslo.
`VALIDATION_IN_RANGE`	Prvek je validní, pokud se jeho hodnota nachází v předaném rozsahu.
`VALIDATION_NOT_EQUAL_TO`	Prvek je validní, pokud jeho hodnota není rovna předanému parametru.
`VALIDATION_PATTERN`	Prvek je validní, pokud jeho hodnota odpovídá předanému

	regulárnímu výrazu.
`VALIDATION_SQL_DATE`	Prvek je validní, pokud jeho hodnota odpovídá datu v SQL formátu.

4.4 Prvky formulářů

4.4.1 Element

Všechny třídy, které reprezentují konkrétní prvky formuláře, dědí od abstraktní třídy Element, která definuje některé základní vlastnosti.

`Element`	`Bs\Registry\Objects\Forms`
`addValidationRule($formValidationConstant, $errorMessage, $value = "")`	Přidá validační pravidlo.
`setClass($class)`	Nastaví css třídu prvku.
`setDisabled($disabled)`	Nastaví prvek na disabled.
`setFillFormWithSubmitedData($fillFormWithSubmitedData)`	Pokud je hodnota true, bude prvek po odeslání formuláře znovu naplněn odeslanou hodnotou.
`setLabel($label)`	Nastaví popisek prvku.
`setTemplateName($templateName)`	Název souboru se šablonou.
`setTemplatePath($templatePath)`	Nastaví cestu k souboru se šablonou formuláře.
`setVisibility($visibility)`	Pokud je hodnota false, bude mít prvek nastavenou viditelnost na hidden

4.4.2 Select

`Select`	`Bs\Registry\Objects\Forms`
`setDisabledItems($disabledItems)`	Přijímá pole s položkami, které není možné vybrat.
`setMultiple($multiple)`	Umožňuje vytvořit prvek, u kterého lze vybrat více hodnot současně.
`setSelectLabel($selectLabel)`	Popisek, který je zobrazen nad seznamem hodnot.
`setSelected($selected)`	Přijímá pole s položkami, které jsou ve výchozím stavu vybrány.
`setSize($size)`	Nastaví velikost prvku.
`setValues($values)`	Nastaví hodnoty prvku. Přijímá pole o struktuře

```
                                Array("value" => "description", "label" =>
                                Array("value" => "description"),)
```

4.4.3 Textarea

Textarea	Bs\Registry\Objects\Forms
setCols($cols)	Nastaví počet sloupců prvku.
setReadonly($readonly)	Prvek není možné editovat.
setRows($rows)	Nastaví počet řádků.
setTransformNl2Br($transformNl2Br)	Pokud je transformace nastavena, tak jsou při získání hodnoty převedeny znaky nového řádku na tag ` `.
setValue($value)	Nastaví hodnotu prvku.

4.4.4 Input

Abstraktní třída, od které dědí všechny další prvky.

Input	Bs\Registry\Objects\Forms
setAutocomplete($autocomplete)	Nastaví možnost automatického doplňování v prohlížeči.
setReadonly($readonly)	Zakáže editaci prvku.

4.4.5 ButtonInput

ButtonInput	Bs\Registry\Objects\Forms
setAlignRight($alignRight)	Nastaví prvku třídu pro zarovnání napravo.
setButtonStyle($buttonStyle)	Nastaví prvku styl reprezentovaný konstantami třídy.
setSize($size)	Nastaví velikost tlačítka.
setValue($value)	Nastaví hodnotu.

4.4.6 HiddenInput

HiddenInput	Bs\Registry\Objects\Forms
setValue($value)	Nastaví hodnotu.

4.4.7 CheckboxInput

CheckboxInput	Bs\Registry\Objects\Forms
setChecked($indexArray)	Přijímá pole s názvy položek, které mají být zaškrtnuty.
setInline($inline)	Zobrazí všechny možnosti na jednom řádku.
setLegendLeft($legendLeft)	Umístí popisek nalevo.
setValues($values)	Nastaví hodnoty prvku. Přijímá pole o struktuře: `$values[value]="legend"`

4.4.8 PasswordInput

PasswordInput	Bs\Registry\Objects\Forms
setMaxlength($maxlength)	Nastaví maximální velikost vstupního řetězce.
setSize($size)	Nastaví velikost prvku.
setValue($value)	Nastaví výchozí hodnotu.

4.4.9 RadioInput

RadioInput	Bs\Registry\Objects\Forms
setChecked($index)	Nastaví položku, která má být vybrána.
setInline($inline)	Zobrazí všechny položky na jednom řádku.
setLegendLeft($legendLeft)	Zarovná popis nalevo.
setValues($values)	Nastaví hodnoty prvku, přijímá pole o struktuře: `$values[value]="legend"`

4.4.10 TextInput

TextInput	Bs\Registry\Objects\Forms
setMaxlength($maxlength)	Nastaví maximální délku vstupního řetězce.
setSize($size)	Nastaví velikost prvku.
setValue($value)	Nastaví výchozí hodnotu.

5 Autentizace uživatelů

Authentication	Bs\Registry\Objects
checkAuthentication()	Kontroluje, jestli se uživatel snaží přihlásit – poté přihlašovací údaje kontroluje proti databázi, pokud sedí, uloží uživatelské jméno do session. Pokud je uživatel již přihlášen, ověří, jestli je uživatel aktivní. Funkce vrací true, pokud je uživatel přihlášen.
getLoggedByPost()	Vrací true, pokud se uživatel právě přihlašuje pomocí formuláře odeslaného metodou POST.
getUser()	Vrací instanci třídy User, která obsahuje data aktuálně přihlášeného uživatele.
logOut()	Odhlásí uživatele.
INDEX_USER_NAME	Konstanta používaná v SESSION a přihlašovacím formuláři.
INDEX_PASSWORD	Konstanta, která se používá jako id prvku v přihlašovacím formuláři.

6 Šablonovací systém

Jmenný prostor:	`Bs\`
`addBlock($nameOfBlock, $tagsArray)`	Předá šabloně pole hodnot, které je poté možné vypsat pomocí iterace. Na blok se v šabloně odkazuje pomocí předaného jména.

`addRecursiveBlock($blocksNames, $tagsArray, $parentsArray)`	Slouží pro výpis rekurzivního typu dat. Přijímá parametry:
	• blocksNames – pole s názvy vnořených bloků (seřazené od vnějších po vnitřní)
	`$blocksNames[] = "název bloku"`
	• tagsArray – pole s názvy proměnných a jejich hodnotami, které obsahuje blok
	`$tagsArray[]["tag_name"] = "value"`
	• parentsArray – pole, které obsahuje rodičovský blok jako index a pole podřízených bloků jako hodnotu
	`$parentsArray[ParentID] = Array("child_ID_1", "child_ID_2")`

`addTag($tag, $value)`	Vloží proměnnou do šablony, ta je poté v šabloně dostupná pod předaným jménem.
`addTags($tagsArray)`	Přijímá pole proměnných, které vloží do šablony. `array (název proměnné => hodnota)`
`addTemplate($templateName)`	Do obsahu stránky vloží šablonovací soubor.
`clear()`	Vyčistí celý obsah stránky, šablona se poté nachází ve stejném stavu, jako po vytvoření nové instance.
`getContent()`	Načte všechny šablonovací soubory, vyplní v nich všechny proměnné, vyhodnotí podmínky a provede překlad, poté vrátí výsledný obsah stránky.
`getTemplatePath()`	Vrací nastavenou cestu k šablonovacím souborům.
`setTemplatePath($templatePath)`	Nastaví cestu k šalonovacím souborům.

Použití podmínky:

```
<!--if test !EMPTY ID 1-->
  <p>Variable "test" is not empty.</p>

  <!--else if test2 = 2 ID 2-->
    <p>Variable "test2" is equel 2.</p>
  <!--end else if ID 2-->
```

```
      <!--else-->
        <p>No condition is true.</p>
       <!--end else-->

    <!--endif ID 1-->
```

Použití proměnné:

```
    <h1>{headingTitle}</h1>
```

Použití bloku proměnných:

```
    <!--start childrensList-->
      <tr class="childrenRow" id="{id}">
         <td>{lastName}</td>
         <td>{firstName}</td>
      </tr>
    <!--end childrensList-->
```

Použití bloku proměnných:

```
<ul>
   <!--start main_menu-->
   <li>
      <a href="{menu_href}" title = "{menu_title}">{menu_name}</a>
      <!--if main_menu_subsection EMPTY ID 100-->
         <ul>
            <!--start main_menu_subsection-->
            <li>
               <a href="{menu_href}" title = "{menu_title}">
                 {menu_name}</a>
               <!--if main_menu_subsection_2 !EMPTY ID 101-->
               <ul>
                  <!--start main_menu_subsection_2-->
                    <li>
                       <a href="{menu_href}" title = "{menu_title}">
                          {menu_name}</a>
                    </li>
                  <!--end main_menu_subsection_2-->
               </ul>
               <!--end if ID 101-->
            </li>
            <!--end main_menu_subsection-->
         </ul>
      <!--end if ID 100-->
   </li>
   <!--end main_menu-->
</ul>
```

Ukázka vstupních dat pro funkci addRecursiveBlock:

```
$blocksNames = array(3) {
  [0]=>
  string(9) "main_menu"
  [1]=>
  string(20) "main_menu_subsection"
  [2]=>
  string(22) "main_menu_subsection_2"
}

$tagsArray = array(7) {
  ["/"]=>
  array(3) {
    ["menu_name"]=>
    string(15) "úvodní strana"
    ["menu_title"]=>
    string(23) "úvodní strana - title"
    ["menu_href"]=>
    string(36) "http://bakalarsky-projekt.localhost/"
  }
  ["novinky"]=>
  array(3) {
    ["menu_name"]=>
    string(7) "Novinky"
    ["menu_title"]=>
    string(0) ""
    ["menu_href"]=>
    string(44) "http://bakalarsky-projekt.localhost/novinky/"
  }...

$parentsArray = array(5) {
  [0]=>
  array(3) {
    [0]=>
    string(1) "/"
    [1]=>
    string(7) "o-firme"
    [2]=>
    string(5) "media"
  }
  ["media"]=>
  array(1) {
    [0]=>
    string(7) "novinky"
  }...
```

Překlad textu:

```
<!--tr t="Hello %s!" p="{name}"-->
<!--tr t="Next %s %s!" p="aa&{name}"-->
<!--tr t="Translate me!" -->

<!--trn t="%s window" pt="%s windows" n="2" p="2"-->
<!--trn t="window %s %s" pt="windows %s %s" n="20" p="10&20"-->
<!--trn t="window" pt="windows" n="20" -->
```

7 Generátor databáze

Framework nabízí programové generování databáze. Ke správné funkčnosti je nutné mít správně nastavený konfigurační soubor. Další podmínkou je, aby každá modelová třída dědila od abstraktní třídy `BasicEntity`. Struktura databázové tabulky se poté definuje přímo v datovém modelu entity.

```
//database table name and column names

const TABLE_NAME = "childrens";
const COLUMN_ID = "id";
const COLUMN_FIRST_NAME = "first_name";
const COLUMN_LAST_NAME = "last_name";

//foreign keyes
const COLUMN_MOTHER_ID = "mother_id";
const COLUMN_FATHER_ID = "father_id";

//many to many table names
const MANY_TO_MANY_FOSTERAGE = "children_fosterage_relation";

public static function defineMySQLTable() {
   $columns = Array(
      new \Bs\DatabaseUtils\TableColumn(
        $name = self::COLUMN_ID,
        $type = \Bs\DatabaseUtils\DataTypes::BIGINT,
        $length = 20, $default = null, $collate = null, $options = null,
        $null = false, $index = \Bs\DatabaseUtils\TableIndexes::PRIMARY,
        $autoIncrement = true, $mimeType = null),
      new \Bs\DatabaseUtils\TableColumn(
        $name = self::COLUMN_FIRST_NAME,
        $type = \Bs\DatabaseUtils\DataTypes::VARCHAR, $length = 100,
        $default = null, $collate = null, $options = null, $null = false,
        $index = null, $autoIncrement = null, $mimeType = null),
      new \Bs\DatabaseUtils\TableColumn(
         $name = self::COLUMN_LAST_NAME,
         $type = \Bs\DatabaseUtils\DataTypes::VARCHAR, $length = 100,
         $default = null, $collate = null, $options = null, $null = false,
         $index = null, $autoIncrement = null, $mimeType = null),
   );

   $options = new \Bs\DatabaseUtils\TableOptions(
      \Bs\DatabaseUtils\StorageEngine::INNO_DB,
      \Bs\DatabaseUtils\Collate::UTF8_CZECH_CI);

   $constraints = Array(
      new \Bs\DatabaseUtils\Constraint($name = "mother",
         $foreignKey = self::COLUMN_MOTHER_ID,
```

```
            $refenceTable = \Bs\Model\Persons\ParentOfChildren::TABLE_NAME,
            $referenceColumn = "id",
            $onUpdate = \Bs\DatabaseUtils\ConstraintOptions::CASCADE,
            $onDelete = \Bs\DatabaseUtils\ConstraintOptions::RESTRICT,
            $optional = true),
        new \Bs\DatabaseUtils\Constraint($name = "father",
            $foreignKey = self::COLUMN_FATHER_ID,
            $refenceTable = \Bs\Model\Persons\ParentOfChildren::TABLE_NAME,
            $referenceColumn = "id",
            $onUpdate = \Bs\DatabaseUtils\ConstraintOptions::CASCADE,
            $onDelete = \Bs\DatabaseUtils\ConstraintOptions::RESTRICT,
            $optional = true),
    );

    $manyToManyTableName = Array(
        Array(self::TABLE_NAME, TextNote::TABLE_NAME),
        self::MANY_TO_MANY_FOSTERAGE => Array(self::TABLE_NAME,
            \Bs\Model\Persons\ParentOfChildren::TABLE_NAME),
    );

    self::$tableDefinition = new \Bs\DatabaseUtils\DatabaseTable(
        self::TABLE_NAME, $columns, $options, $constraints,
        $manyToManyTableName);
}
```