

PHP konference 2008 Brno

Úvod do PHP frameworku CakePHP

1.1 stable

1.2 development

<http://www.cakephp.org>

<http://manual.cakephp.org>

PDF soubor

<http://cakephp.org/files/cakesheet.pdf>

Jiří Knesl

<http://www.knesl.com>

Instalace

PHP 4.3.2+, PHP 5
MySQL 4+, PostgreSQL

app/config/database.php

```
var $default = array('driver' => 'mysql',  
                    'connect' => 'mysql_connect',  
                    'host' => 'localhost',  
                    'login' => 'user',  
                    'password' => 'password',  
                    'database' => 'project_name',  
                    'prefix' => '',  
                    'charset' => 'utf8', );
```

Konzole

V konzoli: cake/console/cake nebo do path (cake)
základní shelly: cake bake, cake acl atd.

Vlastní shell:

vendors/shells/report.php

rozšíření: vendors/shells/tasks/pomocnik.php

```
class ReportShell extends Shell {
    public $uses = array('NejakyModel');
    function main() {
        $data = $this->NejakyModel->findAll('sloupec'=>'hodnota');
        $this->out(sizeof($data));
    }
}
```

Bake

Cake bake

```
cake bake controller  
cake bake model  
cake bake view
```

Slouží pro rychlé generování aplikace. Pokud není nastavená databáze, nabídne konfigurace databáze.

Routování

app/config/routes.php

```
1.1 $Route->map('nase/:action/*', array('controller'=>'cms',  
'action'=>'index', 'první parametr'));
```

```
1.2 Route::connect('nase/:action/*', array('controller'=>'cms',  
'action'=>'index', 'první parametr'), array('action'=>'(index|list|  
edit)'));
```

Bootstrap

`app/config/bootstrap.php`

alternativa je úprava třídy ApplicationController
`/cake/app_controller.php`

Controller - pojmenování

`app/controllers/plurals_controller.php`

- zpracovává logiku
- často zpracovává jeden specifický model, pokud ano, v CakePHP platí konvence, že název controlleru je v množném čísle daného modelu (např. `ArticlesController` použije model `Article`)

Controller II. - actions

Každý controller má tzv.actions.
Actions jsou akce použité pro zobrazení View

```
class ActionController extends ApplicationController
{
    function index()
    {}

    function show($url)
    {}

    function search($query)
    {}
}
```

Controller III. - atributy

```
public $name = 'Articles'; // název controlleru (kvůli PHP4)
public $uses = array('Article', 'Comment'); // použité modely
public $helpers = array('Html', 'Ajax'); // použité helpery
public $components = array('Session', 'Email'); // použité komponenty
public $layout = 'page'; // použitý layout
    často se mění v actions pomocí $this->layout = 'ajax';
    vypnout lze pomocí $this->layout = FALSE;
public $params je array();
    nejčastěji použité klíče:
    form - $_POST a $_FILES
    controller - název controlleru
    action - název action
    pass - /posts/view/?var1=1&var2=2 = „var1=1&var2=2“
    url - /posts/view/?var1=1&var2=2
        = array(„url“=„posts/view“, „var1“=1, „var2“=2)
    data - používají helpery
```

Controller IV. - předání do view

V controlleru: `$this->set('klic_view', 'hodnota');`

Ve view: `<?php echo $klicView; ?>`

- mění podtržítka na camelCase

- lze předat pole, kde klíče se změní na názvy proměnných `$this->set(array('key', 'hodnota'));`

- ukládá v controlleru do `$this->viewVars;`

Controller V. - kontrola view

Určení view:

```
$this->render('action', 'layout');
```

- volaný automaticky

(vypíná se pomocí `$this->autoRender = FALSE;`)

```
$this->layout = 'název layoutu';
```

(`$this->autoRender = FALSE;` vypne taky)

Controller VI. - přesměrování

```
$this->redirect('/', '301', TRUE);
```

- 1. parametr url
- 2. parametr header
- 3. parametr, zda ma zavolat exit;

Controller VII. - Callbacky

function beforeFilter() {} - před action
function afterFilter() {} - po action

Voláno pouze pokud je
\$this->autoRender != FALSE
function beforeRender() {} - před render()

Komponenty

Komponenty jsou logika sdílená mezi controllery.
Komponenty připravené v CakePHP:
Security, Sessions, Acl, Email, Cookies,
Authentication, Request

Vlastní komponenty

```
app/components/my.php  
class MyComponent extends Object {}
```

```
v controlleru public $components = array('My');
```

Metody vlastní komponenty

initialize() {} před beforeFilter controlleru
startup() {} po beforeFilter controlleru

metoda(params) {}

použití v controlleru
\$this->My->metoda('par1');

Sanitace dat

```
Aktivovat lze pomocí: uses('sanitize');  
$sanitize = new Sanitize();
```

```
$sanitize->paranoid(„nechá jen alnum“);  
$sanitize->html(„převéde na entity“);  
$sanitize->sql(„escapuje“);
```

```
$sanitize->cleanArray($pole);  
- slashuje, převádí na entity a další
```

Scaffolding

Aktivuje se v controlleru deklarací `public $scaffold;`
v controlleru.

Jméno sloupce ve výpisech určeno:
`public displayField$ = 'jmeno_sloupce';` v modelu

Lze editovat i šablony.

Propojení controlleru s modelem

Model je volán: `$this->NazevModelu->metoda(params);`

zjednodušení hledání metodou controlleru `postConditions`
`$this->postCondition($this->data, array('>=', 'LIKE'), 'OR');`

1) převede pole z formátu postu: `[Model][sloupec]` na
`[Model.sloupec]`

2) druhý parametr slouží pro určení výrazu pro daný sloupec
(podle pořadí sloupců)

3) spojuje výrazy

Ze zadání: `$this->data = array('A'=>array('x'=>'a', 'y'=>'b'));`
Výsledek: `A.x >= „a“ OR A.y LIKE „b“`

Model

CakePHP používá tzv. Active Record

ORM nástroj, který umí validaci, relace mezi tabulkami, obsahuje callbacky a od 1.2 behaviours.

Složka `app/models/`

Model – umístění a pojmenování

app/models/automobile.php

```
class Automobile extends AppModel
{
    ...
}
```

Model - vytvoření

název jednotné číslo názvu tabulky

```
class Phone extends AppModels { var $name =  
    'Phone'; /* PHP4 */}  
    použije tabulku phones
```

Ize změnit pomocí `public $useTable =
 'telephones';`

žádná použitá tabulka: `$useTable = FALSE;`

Model – další atributy

`$this->useDbConfig` – kterou databázi („default“)

`$this->primaryKey` – primární klíč („id“)

`$this->recursive` – počet tabulek, které se mají
prijoinovávat při dotazích

`$this->order` – defaultní řazení (string, nebo pole)

Model – \$this->id; pozor!

```
foreach ($mojedata as $polozka)
{
    $this->MujModel->save($polozka); // 1 insert, ostatní update
}
```

```
foreach ($mojedata as $polozka)
{
    $this->MujModel->save($polozka); // provede inserty
    $this->MujModel->id = NULL;
}
```

Model - validate

```
public $validates = array('sloupec'=>'/perl_reg/i');  
    VALID_NOT_EMPTY, VALID_NUMBER,  
    VALID_EMAIL, VALID_YEAR
```

Model – validace II.

Metoda `validates()` modelu.
Volána automaticky při `save()`

Poté volán callback `beforeSave()`

pokud vrátí `save()` `FALSE`, nedošlo k uložení

Model – validate III.

Validate podle rails:

<http://bakery.cakephp.org/articles/view/rails-like-data-validation>

ve validates()

```
$this->validates_presence_of('username');  
$this->validates_uniqueness_of('username',array('on'=>'create'));  
$this->validates_length_of('username',array('min'=>3));  
$this->validates_length_of('username',array('max'=>50));  
$this->validates_format_of('email',VALID_EMAIL);  
  
$errors = $this->invalidFields();  
return (count($errors) == 0);
```

Model – automaticky upravované sloupce

Updated
Modified
Created

Model - callbacky

Data jsou v \$this->data

beforeFind()
afterFind()

beforeValidate()

beforeSave() - vrátit TRUE || FALSE
afterSave()

beforeDelete()
afterDelete()

Model – Unit testing

Modely lze testovat:

<http://bakery.cakephp.org/articles/view/testing-models-with-cakephp-1-2-test-suite>

Podpora od 1.2

Používá SimpleTest (

<http://bakery.cakephp.org/articles/view/testing-models-with-cakephp-1-2-test-suite>)

Model – asociace hasOne

```
class User extends AppModel {
public $hasOne = array('Profile' =>
    array(
        'className' => 'Profile',
        'conditions' => '',
        'order' => '',
        'dependent' => true,
        'foreignKey' => 'user_id'
    )
);
}

$this->User->read(NULL, '10');

array(
    'User' => array('id'=>10, 'login'=>'Prihlasimse', 'pass'=>'***'),
    'Profile' => array('id'=>34, 'user_id'=>10, 'address'=>'Lihovar 10')
);
```

Model – asociace belongsTo

```
class Profile extends AppModel {
public $belongsTo = array(
    'User' => array(
        'className' => 'User',
        'conditions' => '',
        'order' => '',
        'foreignKey' => 'user_id'
    )
);
}

$this->Profile->read(NULL, '3');

array(
    'Profile'=>array('id'=>3, 'user_id'=>10, 'address'=>'Praha 1, Hrad'),
    'User'=>array('id'=>10, 'login'=>'Franta', 'pass'=>'***')
);
```

Model – asociace hasMany

```
class Article extends AppModel {
var $hasMany = array(
    'Comment' => array(
        'className' => 'Comment',
        'conditions' => 'Comment.moderated = 1',
        'order' => 'Comment.created DESC',
        'limit' => '5',
        'foreignKey' => 'article_id',
        'dependent' => true,
        'fields' => '',
        'offset' => ''
    )
);
}

$this->Article->read(NULL, '1');
array(
    'Article'=>array('title'=>'Titulek', 'texy'=>'x^2',
        'html'=>'x<sup>2</sup>'),
    'Comment'=>array(
        0=>array('article_id'=>1, 'id'=>10, 'moderated'=>1,
            'created'=>'2005-01-01', 'text'=>'nejaký text'),
        1=>array('article_id'=>1, 'id'=>15, 'moderated'=>1,
            'created'=>'2005-01-10', 'text'=>'nejaký jiný text')
    )
);
```

Model – asociace hasAndBelongsToMany I.

Kardinalita M:N Používá mezitabulku

```
class Post extends AppModel{
public $hasAndBelongsToMany = array(
    'Tag' =>array(
        'className' => 'Tag',
        'joinTable' => 'posts_tags', // konvence je [models1]_[models2]
        'foreignKey' => 'post_id', // konvence [model1]_id
        'associationForeignKey'=> 'tag_id', // konvence [model2]_id
        'conditions' => '',
        'order' => '',
        'limit' => '',
        'unique' => true
    )
);
}
```

Model – asociace hasAndBelongsToMany II.

```
$post = $this->Post->read(null, '2');  
print_r($post);
```

Array

```
(  
    [Post] => Array(  
        [id] => 2  
        [user_id] => 25  
        [title] => Cake Model Associations  
        [body] => Time saving, easy, and powerful.  
        [created] => 2006-04-15 09:33:24  
        [modified] => 2006-04-15 09:33:24  
        [status] => 1  
    )  
  
    [Tag] => Array(  
        [0] => Array ( [id] => 247 [tag] => CakePHP)  
        [1] => Array ( [id] => 256, [tag] => Powerful Software)  
    )  
)
```

Bind modelu

Vytvoří pro 1 dotaz relaci (relace) mezi tabulkami.
Slouží pro snížení náročnosti aplikace.

V controlleru:

```
$this->User->bindModel(  
    array('hasMany'=>array(  
        'Book'=>array(  
            'className'=>'Book'  
        )  
    ))  
);
```

Unbind modelu

Na 1 dotaz odpojí relaci (relace).
Smyslem je nejoinovat nepotřebná data, zvláště při velkých objemech.

V controlleru:

```
$this->Leader->unbindModel(  
    array('hasMany' => array('Follower'))  
);
```

Model – provedení sql dotazů

V modelu:

```
$result = $this->query($sql); // select  
vrací formát více řádků
```

```
$this->execute($sql); // insert, update, delete
```

```
$this->getLastInsertId();
```

Model – formát 1 řádku

Vracená data ve formátu:

```
$result = array(
    'Automobil'=>array(
        'id'           => 5,
        'nazev'        => 'Audi A4',
        'pojisteni_id' => 340004343
    ),
    'Majitel'=>array(
        [0]=>array(
            'id'           => 31,
            'automobil_id' => 5,
            'jmeno'        => 'Josef Novák',
            'vek'          => 50
        ),
        [1]=>array(
            'id'           => 35,
            'automobil_id' => 5,
            'jmeno'        => 'Jana Nováková',
            'vek'          => 45
        )
    ),
    'Pojisteni'=>array(
        'id'           => 340004343,
        'platnost_do'  => '2009-01-02',
        'bonus'        => NULL
    )
);
```

Model – formát více řádků

Vrácená data ve formátu

```
$result = array(  
    [0] => array( ... formát 1 řádku ... )  
    [1] => array( ... formát 1 řádku ... )  
    [2] => array( ... formát 1 řádku ... )  
);
```

Model – základní operace

```
// vytáhne data, ne rekurzivně  
$this->read($sloupce, $id);  
    alternativně:  
    $this->id = 5;  
    $data = $this->read();
```

```
$this->del($id, $cascade = TRUE); // maže – i v  
    závislých tabulkách
```

Model – základní operace II.

```
$this->save($data);
```

očekává formát 1 řádku:

```
(bool) $this->save($data['Model']);
```

pokud obsahuje sloupec `$this->primaryKey`,
ukládá, v opačném případě vkládá

Model – hledání podle sloupce

```
$this->findByNazevSloupce($hodnota);  
(hledá 1 záznam v Model.nazev_sloupce)  
(PHP4: $this->findByNazev_sloupce($hodnota))  
formát 1 řádku
```

```
$this->findAllByNazevSloupce($hodnota);  
(hledá všechny záznamy v Model.nazev_sloupce)  
(PHP4: $this->findByAllNazev_sloupce($hodnota);  
formát více řádků
```

Model – hledání find() a findAll()

```
$this->find($where, $fields, $orderby, $limit,  
           $page, $recursive);  
// formát jednoho řádku
```

```
$this->findAll($where, $fields, $orderby, $limit,  
             $page, $recursive)  
// formát více řádků
```

Find... formát where

A) sql část za where (přijoinované všechny tabulky)

B)

array(„A.sl1“=>1, „A.sl2“=>“> 5“) // spojení and

array(„or“=>

array(„A.sl1“=>1, „A.sl2“=>“> 5“))) // spojení or

array(array(...), „or“=>array(...)) // kombinace

array(„A.sl1“=>array(1, 2, 3)) // A.sl1 IN (1, 2, 3)

Find... formát fields, orderby, limit, recursive

```
$fields = array('A.*', 'B.name', 'C.sl1', 'C.sl2');  
$orders = array('A.poradi', 'B.poradi DESC');  
    $limit = 20;  
    $page = 2;  
$recursive = 2; ( = -1 nic)
```

Model - findNeighbours()

Hledání předchozí a následující hodnoty
(funguje pouze pro čísla a data)

```
$this->findNeighbours($where, $field, $value);
```

V controlleru:

```
$this->findNeighbours(NULL, 'id', 5);
```

Vrací:

```
['prev']['Model']['id'] = 4;
```

```
['next']['Model']['id'] = 6;
```

Model – field a findCount

(string) \$this->field(\$name, \$where, \$order);

Vrací 1 položku db

(int) \$this->findCount(\$where);

Vrací počet vyhovujících dotazu

Model - generateList()

```
$this->generateList($where, $order, $limit, $klic,  
                  $hodnota)
```

Klíč a hodnota se zapisují ve formátu:
{n}.Model.sloupec – např: {n}.Automobile.color

Vrací pole ve formátu:
array('key1'=>'val1', 'key2'=>'val2', ...)

Behaviours v 1.2

V modelu:

```
public $actsAs = array('Tree');  
nebo  
public $actsAs =  
array('Tree'=>array('left'=>'left_id',  
    'right'=>'right_id'));  
  
$this->id = 5;  
$data = $this->children();
```

Behaviours v 1.2

CakePHP 1.2 obsahuje behaviours:

- Tree
- Translalte
- ACL

Aktuálně není dostupná dokumentace.

Layouty

V controlleru: `$this->layout = 'layout';`

1.1: `app/views/layouts/layout.html`

1.2: `app/views/layouts/layout.ctp`

`$title_for_layout` - `$this->pageTitle` v controlleru

`$content_for_layout` – výsledek view

Views

1.1: `V app/views/$controller/$action.html`

1.2: `V app/views/$controller/$action.ctp`

Vrátí se do layoutu.

Elementy

1.1: app/views/elements/nazev.html

1.2: app/views/elements/nazev.ctp

Ve view / layoutu:

```
$this->renderElement('nazev', array('a'=>1,  
                                     'b'=>2));
```

v elementu už jako \$a a \$b

Lze použít helpery.

Helpery

Vlastní v app/view/helpers/nazev.php

```
class NazevHelper extends AppHelper
{
    public $helpers = array('Html');
    function link($title, $url) {
        return $this->output(
            $this->Html->link($title, $url);
        );
    }
}
```

Použití helperů v elementech, views a layoutech

V controlleru:

```
public $helpers = array('Nazev');
```

V šabloně:

```
$nazev->link('text', '/');
```

Form helper v 1.1

```
$html->formTag('/form/submit');  
$html->textarea('Model/sloupec');  
$html->hidden('Model/sloupec2');  
$html->hidden('Model/sloupec3', 'hodnota');  
$html->submit();
```

```
$html->tagErrorMsg('Model/sloupec', 'Text chyby');
```

submit, password, textarea, checkbox, file,
hidden, input, radio

Form helper v 1.2

```
$form->create('Automobile', array('type'=>'file',  
    'action'=>'add'));  
/automobiles/add
```

```
$form->create(NULL,  
array('url'=>'/nekde/nekam'));
```

```
$form->end();
```

Form helper v 1.2

```
$form->input('sloupec');
```

string – text

boolean, tinyint(1) - checkbox

text – textarea

text (password, passwd, psword) – password

date – date výběr

datetime, timestamp – datetime výběr

time – time výběr

Form helper v 1.2

```
$form->input ('soubor', array('type'=>'file'));
```

```
    $form->input ('vyber',  
array('options'=>array(1,2,3,4)));
```

```
options: multiple = TRUE||FALSE (FALSE)  
        maxLength = 10
```

```
div = FALSE; // zda obalovat divem (TRUE)
```

```
label = FALSE // zda přidávat label (TRUE)
```

```
id, rows, cols, empty (prázdný),
```

```
timeFormat (12||24||none), dateFormat (DMY,  
MDY, YMD, NONE)
```

HTML helper v 1.1

```
$html->charset($charset);  
$html->css($název, $rel, $atributy);  
$html->image($cesta, $atributy);  
$html->link($text, $url, $atributy, $confirm,  
           $escape);
```

HTML helper v 1.2

Jako 1.1 navíc:

`$html->meta($title, $url, $atributy, $inline)`

`$html->doctype($doctype)`

`html, html4-strict, html4-trans, html4-frame,`
`xhtml-strict`, `xhtml-trans, xhtml-frame`

`$html->style($poleDat);`

`$html->para($class, $text, $atributy, $escape)`

HTML Table helper v 1.1 a 1.2

```
$html->tableHeaders($nazvy, $tr, $th);
```

```
$html->tableCells($data, $sude, $liche);
```

AJAX helper

Používá Prototype

Podpora drag & drop, auto complete, posílání formu atd.

Pozn.: Existují helpery pro jQuery

Další helpery

Time pro datum a čas, Text pro práci s textem,
Number pro formátování čísel, měny.

Globální funkce

vendor(\$lib1, \$lib2, \$lib3) – načtení z vendors/
uses() - načtení komponenty z cake/libs/
 __(„klíč“) - lokalizace
 a(1, 2, 3) – pole (1, 2, 3)
 aa(„x“, 1, „y“, 2) – hash („x“=>1, „y“=>2)
am(\$pole1, \$pole2, \$pole3...) - array_merge()
 e(\$data) – echo \$data
 pr(\$data) – print_r(\$data)
 low(\$data) - strtolower(\$data)

Lokalizace v 1.2

V controlleru: uses(„L10n“);

app/locale/\$jazyk/LC_MESSAGES/default.po

msgid Klíč

msgstr „Hodnota“

\$this->L10n = new L10n();

\$this->L10n->get('cs');

Configure::write('Config.language', "cs");

__(‘Klíč’);

Cache

```
app/core/config.php  
define ('CACHE_CHECK', true);
```

V controlleru:

```
var $cacheAction = array(  
    'view/23/' => 21600,  
    'view/48/' => 21600  
);
```

Odstranění z cache (ve view apod.)
<cake:nocache>nekešovaný
obsah</cake:nocache>

Log

V controlleru:

```
$this->log('Text zprávy');
```

```
app/tmp/logs/error.log
```

```
$this->log('Text debug', LOG_DEBUG);  
app/tmp/logs/debug.log
```

Konec

Děkuji za Vaši pozornost.