Něco o pclib verze 2.9

Úvod

V následujících příkladech budeme předpokládat, že máme vytvořenou standardní pclib aplikaci používající controllery. Mnoho zmiňovaných věcí funguje i ve starších verzích než je 2.9, někdy už od 2.4.

Konfigurace

Nově lze inicializovat aplikaci také pomocí klíče **pclib.app** v config.php:

```
'pclib.app' => [
	'db' => 'pdo_mysql://...',
	'auth' => false,
	'logger' => false,
	'file-storage' => false,
	'language' => 'cs',
	'default-route' => 'products/list',
	'layout' => 'tpl/layout.tpl',
]
```

Zadáním těchto konfiguračních klíčů můžete vytvořit obvyklým způsobem nakonfigurované služby (ale můžete je vytvořit i původním způsobem v php kódu)

Db vytvoří službu $app->db a připojí se k zadané databázi, **auth** a **logger** zapnou aplikaci s autorizací a logováním atd.

Layout inicializuje šablonu $app->layout a **default-route** je úvodní stránka, která se zavolá, pokud nezadáte žádnou cestu. Zde **ProductsController->listAction()**.

Od verze 2.8 je implicitně zapnutý parametr **tpl-escape = true**, takže html tagy vložené do šablony budou automaticky ošetřeny z důvodu zabezpečení. Pokud potřebujete vložit html, je potřeba v šabloně v sekci elements uvést atribut **noescape** – tedy např. `string CONTENT noescape`.

Od verze 2.8 generuje form pro tlačítka tag **<button>** místo <input> což je v HTML5 oficiálně doporučovaný postup.

Novinky v controllerech

Jednoduchou šablonu můžete zobrazit touto zkratkou:

```
function viewAction() {
	return $this->template('tpl/products/view.tpl', ['title' => 'Fantastic product']);
}
```

Druhý parametr jsou values šablony.

```
function defaultAction() {
	$this->app->error('Stránka nenalezena!', 'alert alert-danger');
}
```

Defaultní akce se volá, pokud požadovaná akce v controlleru neexistuje. Můžete ji využít k zobrazení vlastní obrazovky „Stránka nenalezena“.

```
function getViewContacts($id) {
	return $this->action('contacts/list/companyId:'.$id);
}
```

Metoda **action()** je zkratka, která dovolí zavolat akci libovolného controlleru programově – zde **ContactsController->listAction($id)**

Další zkratkou je metoda **authorize()**. Bez parametrů ověří, zda je uživatel přihlášený, jinak také zkontroluje, zda má příslušné oprávnění. Pokud není přihlášený, přesměruje ho na přihlašovací stránku (**user/signin**) a do session proměnné **backurl** uloží návratovou adresu, ze které bylo přesměrováno.

```
function paymentsAction() {
	$this->authorize('user/payments');
	...
}
```

Funkce redirect(): Kromě přesměrování na routu lze metodu app->redirect() použít i k přesměrování na libovolnou url:

```
$this->redirect(['url' => 'https://www.example.com/info']);

$this->redirect('/');  // Přesměruje na BASE_URL
$this->redirect('/self');  // Přesměruje na aktuální url (hodí se po odeslání formuláře)
```

Databáze

Databáze nyní ukládá do logu dotazy trvající déle než jedna sekunda. Čas můžete změnit nastavením parametru **db->slowQueryLog**, nebo úplně vypnout nastavením db->slowQueryLog = 0

Funkce **insertUpdate()** aktualizuje záznam a pokud klíč neexistuje, záznam přidá. Je to zkratka oproti obdobným funkcím v SQL.

```
$db->insertUpdate($tabulka, $data, ['id']);
```

Poslední parametr je seznam polí, podle kterého se zjišťuje identita.

Dotazy: Jako filtr lze použít i přímo pole.

```
$filter = ['category' => 1, active => 1];
$db->update($tabulka, $data, $filter);
```

„in“ klauzule: Pokud je v dotazu seznam hodnot, lze jako parametr použít i pole.

```
$db->select("select * from A where X in ('{X}')", ['X' => [1,2]]);
```

Vytvoří klauzuli where X in ('1', '2').

Šablony

Nově je možné vložit jednu šablonu do druhé. Například můžete přidat checkbox souhlas s podmínkami do každého formuláře.

Soubor **nejakyform.tpl**:

```
<?elements
include footer file "tpl/partial/standard-footer.tpl"
?>
....
{footer}
```

Můžete vložit výstup nějaké akce Controlleru: action news route "articles/news" – **{news}** pak bude výsledek volání **ArticlesController->newsAction()**

V šabloně lze vytvořit tlačítko nebo link s potvrzovacím dialogem: button delete confirm „Opravdu smazat?“

Lze nastavit globální proměnné, které budou viditelné ve všech šablonách.

```
use pclib\extensions\TplGlobals;
$app->globals = new TplGlobals;

$app->globals->set('image_dir', '/images');
```

Jako druhý parametr set() jde použít callback funkci, která se v šabloně zavolá. Je také možné přidat zcela nové typy template tagů – dejme tomu calendar_input.

Nastavení atributů z kódu: Atributy šablony lze nastavit globálně pro blok nebo celou šablonu.

```
$form->setAttr('noedit', 1); // read-only formulář
```

Hodnoty z bloku: Po odeslání lze načíst odděleně jen ty hodnoty formuláře, které jsou obalené v nějakém bloku.

```
$adresa = $form->getBlock('adresa');
```

Odeslaný soubor lze z formuláře získat takto:

```
$file = $form->getFile('soubor');
print $file;
```

Pokud vypisujeme grid pomocí template tagu **{grid.fields}**, můžeme z php určit které sloupce a v jakém pořadí se v gridu zobrazí.

```
$grid->setFields(['jmeno', 'prijmeni' 'email']);
```

Události

Události byly přepracovány a nyní používají syntaxi známou například z jquery.

```
$grid->on('grid.after-row', function($e) { print 'a'; });
```

Se zavolá po výpisu každého řádku příslušného gridu.

Lze nastavit i událost pro všechny gridy pomocí EventManageru **$app->events**.

```
$app->events->on('grid.before-out', function($e) { print 'a'; });
```

Se zavolá před výpisem každého gridu.

Tree

Třída Tree byla ve verzi 2.6.5 kompletně přepracována. Nyní se inicializuje stejně jako tpl, form nebo grid, tedy cestou k šabloně stromu.

```
$tree = new Tree('tpl/tree.tpl');
```

Když není cesta zadaná, použije se default-tree.tpl. V šabloně jsou bloky **root, folder** a **item**, které definují vnější obal, složku a položku stromu (viz pclib/tpl/default-tree.tpl).

Stejně jako v klasické šabloně můžete nastavovat hodnoty šablony.

```
$tree->values['CSS_CLASS'] = 'pctree';
```

Strom lze načíst z několika zdrojů: z pole, textového souboru nebo databáze. Nejběžnější je načtení z tabulky TREE-LOOKUPS: $tree->load($treeId);

Vypíše se klasicky: $tree->out();

Další možnost je načíst strom z tabulky, kde je stromová struktura uložena pomocí parent_id:

```
$tree->fromQuery("select ID,LABEL,PARENT_ID from STROM");
```

Většina funkcí pro načtení má i odpovídající funkci pro uložení stromu: **importText() / exportText(), fromArray() / toArray(), load() / save()**.

Další funkce jsou například find() pro nalezení uzlu stromu, get() / set() pro přečtení nebo nastavení uzlu, expand() a expandLevel(), které rozbalí strom k nějakému uzlu nebo do nějaké úrovně.

Ukládání souborů - základní

V základním nastavení (bez FileStorage) vytvoříme pole pro nahrávání souborů ve formuláři následovně:

```
input SOUBOR file nosave
```

V tomto případě řešíte ukládání souborů sami, například:

```
$file = $form->getFile('SOUBOR');
file_put_contents('uploaded/' . $form->values['SOUBOR'], $file);
```

Je-li uvedená cesta k adresáři, kam se mají soubory ukládat (into), formulář řeší ukládání, mazání a aktualizování souboru za vás.

```
input SOUBOR file into "uploaded"
```

Funkce **$form->insert()**, **$form->update()** a **$form->delete()** budou teď automaticky aktualizovat všechny soubory ve formuláři. Pokud se formulář ukládá do databáze, musí být v db-tabulce pole SOUBOR, kam se uloží název souboru.

Použití FileStorage

Služba FileStorage slouží k ukládání souborů a document managementu. Umožňuje přiřazovat k libovolným entitám (např. článkům, kurzům, dokladům, uživatelům apod.) neomezené množství souborů a pracovat s nimi jednotně a s lepší organizací. Inicializuje se takto:

```
$app->fileStorage = new pclib\FileStorage('uploaded');
```

kde **'uploaded'** je cesta k adresáři, kam budou soubory ukládány. Implicitně jsou ukládány do podadresářů uploaded/rok/měsíc/, které FileStorage vytváří.
Tedy např. uploaded/2023/10/product_18urYQjU.jpeg (název obsahuje hash, aby ho nebylo možné uhádnout). Způsob ukládání lze ale změnit.

Pak input SOUBOR file ukládá prostřednictvím FileStorage do adresáře **uploaded**. Informace o souboru (velikost, typ, datum, cesta k souboru) je zároveň uložena do db tabulky FILESTORAGE. Tedy zavolání $form->update('clients', 1); uloží pro klienta 1 soubor SOUBOR.

(Filestorage lze ale využívat i samostatně a nezávisle na formulářích.)

Se soubory můžeme pracovat pomocí funkcí **getFile(), setFile(), deleteFile()** pro jeden soubor, nebo **getFiles(), setFiles(), deleteFiles()** které vrátí, nastaví nebo smaže všechny soubory přiřazené konkrétní entitě. **postedFiles()** vrátí soubory odeslané POSTem z formuláře.

Příklad:

```
$fs = $app->fileStorage;

$fs->setFiles(['clients', 1], $files); //Uložíme soubory ke klientovi s id=1
$files = $fs->getFiles(['clients', 1]); //Načteme všechny soubory klienta #1 jako pole
$file = $fs->getFile(['clients', 1, 'SOUBOR']); //Načteme soubor SOUBOR klienta #1
```

$file je záznam z tabulky FILESTORAGE. $file['FILEPATH'] je cesta k souboru, $file['ORIGNAME'] je původní jméno souboru, $file['MIMETYPE'] je typ souboru atd.
Entita, ke které soubory patří, se zadává jako dvouprvkové pole [typ_entity, id_entity] např. ['clients', 1].
Konkrétní soubor lze adresovat jako pole [typ_entity, id_entity, id_souboru] anebo jen pomocí jeho id z databáze – např. **$fs->getFile(1)**;

Chceme-li soubor zobrazit, můžeme tak učinit příkazem **$fs->output($id);** kde $id je ID z tabulky FILESTORAGE nebo lépe **$fs->output(['HASH' => $hash]);** (HASH z tabulky filestorage).
To je lepší než se v prohlížeči přímo odkazovat na cestu k souboru a neumožní to uživateli změnou url zobrazit soubory, ke kterým nemá oprávnění.

API třídy FileStorage bylo ve verzi 2.8.5 aktualizováno.

Aplikační parametry

Některé parametry Nastavení (Možnosti) aplikace je potřeba měnit a je dobré mít správcovské rozhraní, kde je můžeme nastavovat. K tomu slouží tabulka APP_PARAMS a sekce aplikačních parametrů v padminu.
V aplikaci je můžeme použít následujícím způsobem:

```
use pclib\extensions\AppParams;
$app->params = new AppParams;

$param = $app->params->get('nazev_parametru');
```

Různé

Některé zajímavé funkce přidané od verze 2.1:

```
$user->changePassword("tajne"); //Změna hesla v objektu AuthUser
$user->hasRole("admin"); //lze otestovat jestli má uživatel příslušnou roli
$grid->exportExcel(); //Vytvoří z gridu csv soubor s nastavením pro Excel a otevře ho v prohlížeči
$form->dbSync("TABULKA"); //Nastaví maxlength inputů formuláře tak, aby nepřekročily velikost pole databáze.
// Lze zadat i v šabloně: class form ... table "TABULKA"
```

Pro práci s aplikačními session proměnnými lze použít funkce objektu App. Např:

```
$app->setSession("message", "Hello");
if ($app->getSession("message")) {
        print $app->getSession("message");
        $app->deleteSession("message");
}
```

Novinky v padmin

V padmin 2.1 přibyla například možnost filtrovat uživatele přidané za poslední měsíc a uživatele nepřihlášené déle než rok, na formuláři uživatele pak například kopírovat uživatele nebo přihlásit se jako zvolený uživatel. Dál byla přidána nová sekce pro zadávání aplikačních parametrů.
Bylo přidáno šifrování hesel **bcrypt-md5**, vhodné, pokud už máte uživatele s heslem šifrovaným md5. Migrace v padminu hash převede na bezpečnější.

V config.php:

```
$config['pclib.auth'][ 'algo'] = 'bcrypt-md5'; //další moznosti: md5,bcrypt
```

Zadáním padmin/?r=install/servicejobs v adresní řádce nainstaluje do padmin servisní příkazy včetně JobMigrateHash (od verze padmin 2.2.2).