

Підпрограми calls.pm

Глобальні змінні

- `$ses::ajax` - встановлений, якщо запит є ajax.
- `$ses::cookie` - поточні куки, наприклад `$ses::cookie→{debug}`.
- `$ses::http_prefix` - 'http://' або 'https://'.
- `$cfg::img_url` - '<https://nodeny.com.ua/i>'.
- `$ses::ip` - ip адміна/клієнта, який залогінився в веб-інтерфейсі.
- `$ses::server` - ім'я сервера витягнутого з адресного рядка, наприклад, `nodeny.com.ua` в <https://nodeny.com.ua/cgi-bin/stat.pl>.
- `$ ses:: t` - поточний час, unix timestamp.
- `$ses::day_now` - поточний день.
- `$ses::mon_now` - поточний місяць.
- `$ses::year_now` - поточний рік.
- `$ses::auth` - посилання на дані поточної авторизації в адмінці:
 - `$ses::auth→{auth}` встановлений, якщо адмін/клієнт авторизований.
 - `$ses::auth→{uid}` - id адміна/клієнта
 - `$ses::auth→{role}`: 'admin' - авторизований адмін, 'user' - клієнт
 - `$ses::auth→{ses}` - рядок-сесія
 - `$ses::auth→{trust}` - чи довірена сесія
- `$ses::unikey` - ключ до даних у таблиці `webses_data`, які будуть додані до поточних даних, переданих браузером.

У будь-який модуль, крім авторизації, можна потрапити лише будучи авторизованим. Авторизація буває довіреною та недовіреною. Якщо джерело авторизації надійне, наприклад, авторизація за логіном та паролем, він встановлює параметр `trust` в 1. Небезпечні функції слід дозволяти тільки при довіреному з'єднанні:

```
$ses::auth->{trust} or Error('З'єднання не довірене. Необхідно перелогінитися');
```

До клієнтської статистики може потрапити не тільки клієнт, а й адміністратор від його імені, тому в деяких випадках бажано виводити різну інформацію для адміна та клієнта, а також фіксувати, що саме адміністратор виконав дію:

```
if( 1 + 1 > 3 )
{
    Error( Adm->id? 'Паніка! Perl неправильно вважає' : 'Сервіс тимчасово недоступний');
}
# Поповнимо рахунок клієнта, при цьому Pay_to_DB сама проставит автора
Pay_to_DB(uid=>$uid, cash=>10, category=>1);
```

POST/GET параметри

Отримати параметри, надіслані через браузер, можна за допомогою `ses::input`:

<http://xxx?a=test&b=0&uid=33>

```
if( ses::input('a') eq 'test' && ses::input_exists('b') )
{
    # Умова спрацює т.к. в url a = test і параметр b присутній
    # Параметр uid про всяк випадок примусово приведемо до цілого числа
    my $uid = ses::input_int('uid');
    $uid or Error('Введіть ціле число, що не дорівнює нулю');
}
```

Хеш усіх переданих параметрів зберігається в `$ses::input_orig`:

```
# Якщо параметр yes не існує або не встановлено,
# відредагуємо самі на себе і пошлемо всі параметри + встановимо yes
ses::input('yes') or url->redirect( { %$ses::input_orig, yes=>1 } );
```

Існує альтернативний спосіб передачі параметрів - дані записуються в базу даних таблицю [webses_data](#), доступ до яких здійснюється за ключом `unikey` таблиці.

Взагалі через дану таблицю можуть передаватися будь-які дані, а передача параметрів від браузера - одна з фіч. На прикладі покажемо

навіщо потрібний такий механізм.

Припустимо, клієнт надсилає дані. Модуль їх перевіряє. Знаходить помилку. Якщо вивести помилку, клієнт буде роздратований тим, що ввів багато даних і йому знову знадобиться їх вводити. Тому передані дані записуються в `webses_data` і відбувається редирект на сторінку введення, при цьому параметр `_unikey` встановлюється значення поля `unikey` збережених даних.

Перед завантаженням модуля, якщо `calls.pm` бачить параметр `_unikey`, він автоматично витягує з БД зв'язану з ключем рядок і дешифрує її змінну `$ses::data`. Далі, якщо існує ключ `$ses::data→{-input}`, то `calls.pm` «дає вигляд», що браузер надіслав POST/GET дані, що знаходяться `$ses::data→{-input}`.

Найчастіше ці події відбуваються прозоро, тобто. не потрібно думати про ключ `-input`. Наприклад, редирект завжди здійснюється через `webses_data`:

```
$Url->redirect(a=>'main', uid=>15);
```

В даному випадку відбудеться запис `{a=>'main', uid=>15}` у `webses_data`, після чого відбудеться перехід `http://xxx/?_unikey=uuuuuu`. Тут `uuuuuu` – ключ до даних, записаних у таблицю.

Якщо ми вже заговорили про редирект, то відзначимо 2 корисні його параметри `-made` і `-error`. Перший призначений для того, щоб вивести повідомлення вгорі сторінки після редиректу, при цьому якщо буде встановлено `-error`, то червоним кольором:

```
$Url->redirect(a=>'main', -error=>1,  
-made=>'Помилкова операція. Редиректимосся на титульну сторінку');
```

Підпрограми модуля main

ToLeft, ToRight, ToTop

Виводять дані в ліву, праву і відповідно у верхню частину html-сторінки, що формується.

```
ToLeft 'Текст зліва';
```

MessageBox виводить повідомлення у рамці

```
ToLeft MessageBox( 'Повідомлення 1');
ToRight MessageBox( 'Повідомлення 2');
```

MessageWideBox виводить повідомлення у рамці, розтягнутій по ширині

```
Show Message
WideBox( 'Повідомлення');
```

Box виводить темплейт box.html. Усі параметри необов'язкові

- title - заголовок повідомлення
- msg - повідомлення
- wide - якщо встановлений, то рамка розтягується по ширині
- css_class - css class повідомлення

```
Show Box(
    title => 'Заголовок',
    wide => 1,
    msg => 'Текст повідомлення всередині рамки',
    css_class => 'error txtpadding',
);
```

WideBox теж, що й Box, але параметр wide встановлений в 1

```
Show WideBox();
```

Menu

Menu теж, що і MessageWideBox, але повідомлення має css class `navmenu`, що наказує виводити гіперпосилання на всю ширину блоку, одне під одним.

```
ToLeft Menu(
```

```
url->a( 'Список клієнтів', a=>'users' ).  
url->a( 'Статистика трафіку', a=>'traf_log' ).  
url->a( 'Google', -base=>'http://google.com' )  
);
```

Center виводить інформацію відцентрованої по горизонталі

```
Show Center MessageBox('Center');
```

Відмінність від css h_center у цьому, що дозволяє центрувати як текст.

Error виводить повідомлення про помилку та завершує виконання скриптів

```
1 > 2 && Error('1 > 2 !!!');
```

Error_ теж, що й Error, але повідомлення обробляється підпрограмою _()

```
$lang::hello_msg = 'Здрастуйте [filtr|bold], на вашому рахунку [bold] $';  
Error_($lang::hello_msg, 'адміністратор', 1000);
```

ErrorMess теж, що й Error, але виконання скрипту не припиняється

```
ErrorMess('Щось не так!');  
Error('Точно щось не так...');
```

Pay_to_DB

Pay_to_DB створює запис у таблиці paus за вхідними параметрами:

- uid - id клієнта, з яким пов'язана запис чи 0, а то й пов'язана. Необов'язковий параметр за промовчанням 0.
- cash - сума, або 0, якщо платіж не фінансовий. Необов'язковий параметр за промовчанням 0.
- reason - поле reason. Необов'язковий параметр.

- comment - поле comment. Необов'язковий параметр.
- time - timestamp платежу. Необов'язковий параметр за промовчанням поточний час.
- category - категорія. Обов'язковий параметр.

Повертає 1 у разі успішного запису. У параметрах запису автоматично встановлюється автор адмін або клієнт залежно від авторизованого. Також встановлюється IP.

Якщо запис фінансовий, то має бути виконаний у транзакції, наприклад:

```
Db->begin_work або Error($lang::err_try_again);
my $rows1 = Db->do("UPDATE users SET balance=balance+(?) WHERE id=?", $money, $uid);
my $rows2 = Pay_to_DB(uid=>$uid, cash=>$money, category=>1);
if( $rows1 < 1 || $rows2 < 1 || !Db->commit )
{
    Db->rollback;
    Error($lang::err_try_again);
}
```

Тут Db->begin_work перемикає Db на режим транзакцій. Починаючи з цього моменту, sql update/insert будуть виконуватися, але не фіксуватися в БД поки не буде виконаний Db->commit. У разі перевіряється, що обидва запити було виконано успішно, тобто. якщо хоча б один із запитів повернув менше одиниці - виконується відкат Db->rollback. Відкат гарантує, що жоден із запитів не буде зафіксовано у БД. Також він вимикає режим транзакцій.

Get_usr_info

Отримує дані клієнта щодо його id. У разі успіху повертає посилання на хеш з даними, інакше повідомлення про помилку.

```
my $ info = Get_usr_info(15);
ref $info або Error $info;
Show 'ПІБ клієнта з id=15: '.v::filtr($info->{fio});
```

Підпрограма повертає як основні дані, а й додаткові поля, і навіть список ір, які належать клієнту.

```
debug('pre', $info);
```

_()

Підпрограма, що складається з одного символу підкреслення, вставляє в заданий рядок параметри:

```
Show _('[div][p h_center][div bold h_right]', 'Текст1','Текст2','Текст3');
```

Перетворюється на:

```
<div>Текст1</div><p class='h_center'>Текст2</p><div class='bold h_right'>Текст3</div>
```

Як видно, квадратні дужки вказують на вставку чергового параметра у поточне місце. При цьому перше слово у квадратних дужках вказує на тег. Є кілька зарезервованих слів, які не вказують на тег, наприклад, `filtr` - ескейпиту html-спецсимволи в параметрі. `trim` - видаляє прогалини по краях параметра. Крім того, вертикальна характеристика дозволяє вставляти кілька керуючих послідовностей:

```
Show _('Ви ввели [filtr|trim|span bold]', 'xxx');
```

Перетворюється на рядок «Ви ввели xxx»

Таблиці

perl код

```
# Створюємо таблицю
my $tbl = tbl->new(-class=>'td_wide');
# Перший рядок
```

```
$tbl->add( '*', 'lll', 'комірка 1','комірка 2','комірка 3' );  
# Другий рядок  
$tbl->add( '*', 'lll', 'комірка 1','комірка 2','комірка 3' );  
# Виводимо таблицю  
Show $tbl->show;
```

Методом new створюється нова порожня таблиця. Вхідні параметри:

```
-class : css class таблиці  
-row1  : css class першого ряду  
-row2  : css class другого ряду
```

Параметри не є обов'язковими. Можуть бути взяті з іншого:

perl код

```
my $tbl1 = tbl->new( -class=>'td_wide td_ok' );  
# Class для таблиці 2 буде 'td_wide td_ok'  
my $tbl2 = $tbl->new(-row1=>'row5');
```

Приклади класів для таблиць

```
td_wide  : таблиця максимальної ширини  
fade_border : осередки відокремлюються напівпрозорими лініями  
pretty   : якщо в таблиці буде мало рядків, то вони будуть більшими за висотою  
td_ok    : оптимальний паддинг для осередків таблиці  
td_tall  : високі рядки  
td_medium : рядки середньої висоти  
td_wide  : широкі осередки  
td_narrow : вузькі осередки
```

Рядки додаються методами add (наприкінці таблиці) і ins (на початок таблиці.). Формат параметрів:

- css ряду, що вставляється

- рядок - карта осередків
- осередок 1
- осередок 2
- ...

Якщо в css буде символ зірочка - це вказівка дати ряду css з параметра -row1, після чого значення в -row1 і -row2 обміняються місцями. У такий спосіб можна організувати «зебру».

Карта осередків - це рядок, який описує вирівнювання кожного осередку:

```
l : вирівнювання по лівому краю
r : з правого
c : по центру
L : по лівому та 2 осередки об'єднуються в один
R : праворуч і 2 осередки об'єднуються в одну
C : по центру та 2 осередки об'єднуються в один
E : по лівому та 3 осередки об'єднуються в один
3 : по центру та 3 осередки об'єднуються в один
4 : по центру та 4 осередки об'єднуються в один
..
9: по центру та 9 осередків об'єднуються в одну
t : по центру та вертикальне вирівнювання top
T : 2 осередки в одну, по центру та вертикальне вирівнювання top
пробіл : порожній осередок
```

perl код

```
my $tbl = tbl->new( -class=>'td_wide td_ok' );
$tbl->add( '*', 'rcl', 'рівняння праворуч', 'центр', 'ліво');
$tbl->add( '*', 'rL', 'текст у 1й комірці', 'комірка 2 і 3 об'єднана в 1');
$tbl->add( '*', '3', 'текст центром і всі осередки об'єднані в одну');
Show $tbl->show;
```

Якщо будь-яка комірка буде у квадратних дужках - це відключить фільтрацію html-спецсимволів. Якщо ви відображаєте дані, надіслані

клієнтом, ви не повинні відключати фільтрацію.

perl код

```
$tbl->add( '*', 'rl', 'ви надіслали рядок', ses::input('str'));
$tbl->add( '*', 'rl', 'поле введення', [ v::input_t(name=>'str') ]));
```

Автоматично ведеться облік рядків та стовпців таблиці. Метод rows дозволяє отримати поточну кількість рядів. Якщо при вставці рядків з'ясовується, що осередок менше ніж у попередньому, то йде автоматичне розширення останнього шляхом об'єднання з рештою:

perl код

```
$tbl->add( '*', 'll', 'Здрастуйте', 'Станіслав');
осередок буде з'єднаний з останньою т.к. у попередньому рядку 2 стовпці
$tbl->add( '*', 'l', 'сьогодні все ок');
# return не спрацює т.к. у таблиці 2 рядки
$tbl->rows < 1 && return;
```

Альтернативний спосіб вставки рядків дозволяє надавати css клас персонально кожному осередку (а не тільки всьому рядку в цілому). Крім того, для кожного осередку вказується заголовок. Кожен рядок необхідно обернути у квадратні дужки:

perl код

```
$tbl->add( '*', [
    [ 'bold h_left', 'Заголовок 1й колонки', 'Увага' ],
    [ 'h_right', 'Заголовок 2й колонки', 'усім' ],
]);
```

У результаті буде вставлено рядок із двома осередками: у першому вирівнювання по лівому краю, у другій - по правому. У лівій виділений (клас bold) текст «Увага», у правій текст «усім». Коли таблиця буде виведена, то її шапці у першому осередку буде текст «Заголовок 1й колонки», у 2й - «Заголовок 2й колонки».

Package Url

Призначений на формування гіперпосилань.

Методи

```
new : створення нового об'єкту
a   : рендеринг url-об'єкта в гіперпосилання
post_a : рендеринг у гіперпосилання, при натисканні на яке
        дані будуть надіслані post-методом
form : рендеринг url-об'єкта у форму
url  : отримання url з об'єкта
redirect: редирект
```

Якщо параметр **не** починається зі знака тире - це параметр url-a:

perl код

```
my $url = url->new( a=>1, b=>2, -class=>'nav', -style=>'color:#000' );
Show $url->a('Текст посилання');
```

Буде перетворено на `Текст посилання`

Одне посилання можна створити на основі іншого, можна перевизначити параметри і т.д.

perl код

```
my $url = url->new( a=>1, b=>2, -class=>'nav', -style=>'color:#000' );
my $url2 = url->new( b => 3); #?a=1&b=3
$url2->{a} = 'demo'; #?a=demo&b=3
Show $url2->a('OK', b=>4); #?a=demo&b=4
$url->{b} = undef;
```

```
Show $url2->a('OK'); #?a=demo
```

У всіх параметрах ескейп html-специфічних символів. Квадратні дужки відключають це, найчастіше необхідно для тексту посилання:

perl код

```
my $url = url->new(-class=>'nav');  
Але краще це зробити через стиль або клас  
Show $url->a( [ '<b>Bold text</b>' ] );
```

У формах порядок параметрів відрізняється від гіперпосилання, спочатку параметри, потім вміст форми:

perl код

```
my $form = _(['p][p h_center][p h_center]',  
  'Введіть ціле позитивне число:',  
  v::input_t(name=>'num'),  
  v::submit('Далі')  
);  
my $form = $Url->form(  
  a=>'module_name', param2=>'test', $form  
);
```

Параметр `-ajax` автоматично конвертується на `-class='ajax'`. При цьому JS NoDeny усі посилання, у яких присутній такий клас, робить «аяксовими» - при натисканні на такі посилання не завантажується нова сторінка, але йде http-запит на сервер, а відповідь обробляється скриптом `nody.js` у браузері.

Package Db

Призначений для роботи з базою даних. Основний метод – sql.

2 формати виклику

- Db->sql(sql, параметри для плейсхолдерів)
- Db->sql({ параметри})

Якщо необхідна вибірка лише одного рядка:

```
my %p = Db->line(параметри);
```

Хеш %p порожній якщо

- Порожня вибірка
- сталася помилка (невірний sql, дисконнект БД,...)

Щоб уточнити: DB->ok повертає 1, якщо не було помилок.

Вибір одного рядка

```
my %p = Db->line("SELECT * FROM users WHERE id=? AND grp=?", $id, $grp);  
Show %p? "{$p{name}, $p{fio}" : Db->ok? 'порожня вибірка' : 'внутрішня помилка';
```

Вибірка кількох рядків

```
my $db = Db->sql("SELECT id, name FROM users WHERE field=?", $unfiltered_field);  
while( my %p = $db->line )  
{  
    Show "{$p{id} = $p{name}<br>";  
}
```

Вибір декількох рядків з іншим форматом виклику

```
my $db = Db->sql(  
    sql => "SELECT * FROM tbl WHERE field=? AND val=?",  
    param => [ $filed, $val ],  
    comment => 'Вибірка номер 2',  
);  
while( my %p = $db->line ) { ... }
```

UPDATE/INSERT

```
my $rows = Db->do("UPDATE websessions SET uid=?, role=? WHERE ses=? LIMIT 1", $id, $role, $ses);  
$rows>0 or Error('Помилка!'); не робіть $rows or Error() т.к. rows може = -1
```

Виконання кількох запитів у транзакції

```
Db-> do_all (  
    [$sql1, $param1, $param2 ],  
    [$sql2, $param3 ],  
);
```

Перевіряється, що кожен запит торкнувся як мінімум 1 рядка. Увага! Якщо запит виконався, але не торкнувся жодного рядка (жоден збіг за умовою WHERE), то буде відкат транзакції.

Додаткові методи

- disconnect - розриває з'єднання з БД
- connect - здійснює з'єднання з БД
- begin_work - старт транзакції
- commit - коміт транзакції
- rollback - відкат транзакції
- ok - чи був успішний останній виконаний sql
- rows - рядків торкнувся останній виконаний sql
- filtr - еcranування небезпечних символів sql

From:
<https://ndp.pp.ua/> - my NoDeny Wiki

Permanent link:
<https://ndp.pp.ua/doku.php/nodeny/docs/calls.pm>

Last update: **15/05/2023 12:46**

