

# DotKernel

- Cum se updateaza o noua versiune
- Cum se scrie codul
- Observatii

# Cum se updateaza o noua versiune

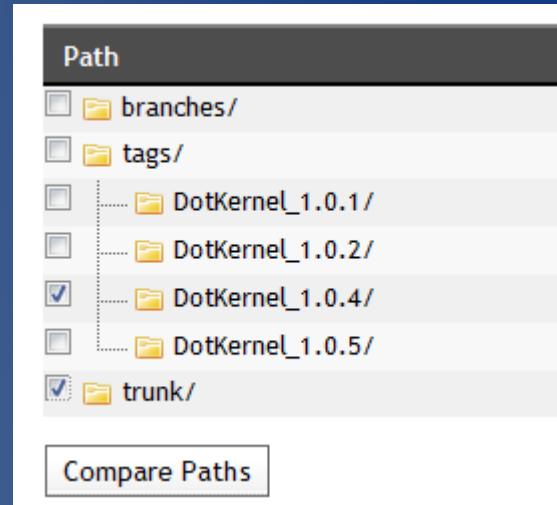
- <http://www.dotkernel.com/releases>
- Change Log – ce fisierele s-au modificat de la ultima versiune
- Manual - <http://www.dotkernel.com/manual/tips>
- Documentatie:  
[http://www.dotkernel.com/manual/phpdoc\\_1.0.5](http://www.dotkernel.com/manual/phpdoc_1.0.5)

# Cum se updateaza o noua versiune

- WEBSVN - <http://websvn.dotkernel.net>
- /trunk – varianta curenta
- /tags – versiunile lansate (releases)

# Cum se updateaza o noua versiune

- Comparare versiuni
- - sters
- + adaugat



The screenshot shows a code editor displaying a diff between two versions of a PHP file. The title bar indicates the file path is '/trunk/controllers/admin/indexController.php'. Below it, a status bar shows '46,9 → 46,10'. The code itself is as follows:

```
/**
 * From this point , the control is taken by the Action specific controller
 * call the Action specific file
 * call the Action specific file, but check first if exists
 */
require(CONTROLLERS_PATH . '/' . $requestModule . '/' . $requestController . 'Controller.php');
$actionControllerPath = CONTROLLERS_PATH . '/' . $requestModule . '/' . $requestController . 'Controller.php';
!file_exists($actionControllerPath) ? $dotKernel->pageNotFound() : require($actionControllerPath);
```

The code is annotated with several colored highlights: the first two lines ('From this point' and 'call the Action specific file') are orange, the third line ('call the Action specific file, but check first if exists') is green, and the entire block below ('\*/' and the require statements) is light orange.

# Cum se updateaza o noua versiune

- Compararea a doua fisiere

**COMPARE REVISIONS**

This comparison shows the changes necessary to convert path

 /tags/DotKernel\_1.0.4/index.php

Rev HEAD → Rev HEAD

↔ Reverse comparison

Compare Path:  Rev

With Path:  Rev

# Cum se scrie codul

- Fiecare metoda va avea comentariu de definire:

```
/**  
 * Return the user Ip , whatever the server are set  
 * @access public  
 * @static  
 * @return string  
 */  
public static function getUserIp(){...}
```

- Comentariu in interiorul metodei:

case 'auth':

```
// validate the authorization request paramethers  
$validate = Dot_AuthorizeUser::validateLogin($_POST['username'],  
                                              $_POST['password'],  
                                              $_POST['send']);  
if(!empty($validate['login']) && empty($validate['error']))  
{  
    // login info are VALID, we can see if is a valid user now  
    $user = $frontendUser->checkLogin($validate['login']);  
    ... }
```

# Cum se scrie codul

- Unele blocuri de cod se vor pune in functii private (chiar daca se apeleaza o singura data)
  - ex. listare tari, judete
  - mai usor de citit codul
  - codul e mai ordonat, mai curat

# Observatii

- Nu folositi variabila GET in URL
  - .../index.php?a=1&b=2
- Nu folositi metoda GET in formulare, ci POST
  - <form method="GET" action="/user/register">
- Folositi – (minus) in loc de \_ (underscore) acolo unde este nevoie