



(<http://github.com/oscardias>)



(<https://twitter.com/oscardias>)



(<https://facebook.com/oscardias>)



(<http://google.com/+OscarDias>)



(<http://linkedin.com/company/oscardias>)



(<http://oscardias.com>)

Search



Home (<http://oscardias.com/br/>)

Projetos ▾

Sobre (<http://oscardias.com/br/sobre/>)

CodeIgniter (<http://oscardias.com/br/category/desenvolvimento/php/codeigniter/>)



(http://oscardias.com/br/wp-content/uploads/2012/05/thumb_codeigniter1.png)

Criando um App usando CodeIgniter (Parte 2): Instalação

👤 Oscar Dias (<http://oscardias.com/br/author/oscardias/>) 📅 28 de maio de 2012 👁 Criando um App usando CodeIgniter ▾ 17 Comments
(http://oscardias.com/br/desenvolvimento/php/codeigniter/criando-um-app-usando-codeigniter-parte-2-instalacao/#disqus_thread)

Neste artigo vou mostrar como configurar nossa instalação do CodeIgniter e começar o nosso app. Para isso vamos definir duas tabelas no banco de dados e criar uma lógica para fazer a “instalação” do aplicativo. Esse artigo é parte do “Criando um App usando CodeIgniter” e continua após a Introdução.

Introdução

Vamos começar esse artigo com um pouco de planejamento. Nosso objetivo é construir o aplicativo Simple Task Board (<http://oscardias.com/br/desenvolvimento/php/codeigniter/projeto-em-php-simple-task-board/>) do zero. Para atingir nosso objetivo, essa é nossa agenda para as próximas semanas:

1. Introdução (<http://oscardias.com/br/desenvolvimento/php/codeigniter/criando-um-app-usando-codeigniter-parte-1-introducao/>) (já publicado)
2. Instalação: instalação do CodeIgniter, banco de dados (usuários e sessões) e procedimento de instalação
3. Login: login, criação da biblioteca Template e layout (menus)
4. Usuários: editar/adicionar/remover usuários
5. Projetos: banco de dados (projetos), visualizar tarefas do projeto (quadro de tarefas), editar/adicionar projeto
6. Tarefas: banco de dados (tarefas), editar/adicionar tarefas

Essa é a primeira versão desse aplicativo. Portanto, como eu for atualizando, vou adicionando novos artigos para explicar o que foi feito. Enfim, vamos começar.

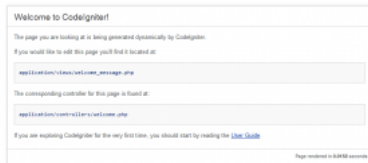
Instalação

A primeira coisa que precisamos fazer é baixar a última versão do CI (CodeIgniter). Extraia o conteúdo do ZIP no diretório que você desejar (eu vou utilizar `localhost/task_board_tutorial`). Você precisará de um servidor instalado na sua máquina local, mas eu suponho que você já saiba isso (se você precisar de ajuda, avise nos comentários). A próxima coisa é remover o diretório `user_guide`. Nós não precisamos dele porque podemos acessar online. Isso é o que você deve ver:

- application
- system

- index.php
- license.txt

Se você for para http://localhost/task_board_tutorial essa página aparecerá:



(<http://oscardias.com/wp-content/uploads/2012/05/Welcome-to-CodeIgniter.png>)

Agora vá para o diretório `application/config`. Vamos atualizar alguns arquivos aqui:

config.php

Como você pode imaginar, a maior parte das configurações acontece aqui. Vamos atualizar os seguintes valores:

```
1 | $config['base_url'] = 'http://localhost/task_board_tutorial/' (http://localhost/task_board_tutorial/);?
```

O valor `base_url` será utilizado pelo CI para criar os links no nosso aplicativo. Então, quando você criar um link utilizando a função `anchor()`, este valor será utilizado para criar o link completo.

```
1 | $config['index_page'] = '';
```

Remova o valor da página de índice. Se você deixar o valor `index.php`, seus links vão ficar como `http://localhost/task_board_tutorial/index.php/controller/method` ao invés de `http://localhost/task_board_tutorial/controller/method`. Mas para fazer os links funcionarem sem o `index.php`, também precisamos de um arquivo `htaccess`. Então vá para o diretório raiz e crie um arquivo `.htaccess` com o seguinte conteúdo:

```
1 | RewriteEngine On
2 | RewriteCond %{REQUEST_FILENAME} !-f
3 | RewriteCond %{REQUEST_FILENAME} !-d
4 | RewriteRule ^(.*)$ /task_board_tutorial/index.php/$1 [L]
```

Você precisa do `mod_rewrite` do Apache para isso. Não vou falar sobre Apache aqui então, se você tiver algum problema, me avise nos comentários. Volte para o arquivo `config.php`.

```
1 | $config['encryption_key'] = 'j4Hg4sptw39HTE26d0IncYT67dH6yfi2';?
```

Como vamos utilizar a classe `Session` que vem com o CI (para controlar as sessões), nós precisamos definir o valor `encryption_key`. Precisa conter 32 caracteres e ser o mais randômica possível. Você pode, e deve, criar a sua própria string. Veja a referência oficial (http://codeigniter.com/user_guide/libraries/encryption.html).

```
1 | $config['sess_cookie_name'] = 'stb_session';
2 | $config['sess_use_database'] = TRUE;
```

Vamos modificar o valor do cookie de sessão para `stb_session` (stb como em Simple Task Board :D). Nós vamos utilizar o banco de dados para controle de sessão, então modifique `sess_use_database` para `TRUE`. A tabela de banco de dados ficará com o mesmo nome, então não precisamos mudar. Você pode modificar a duração da sessão como preferir (`sess_expiration`). Isso é tudo que precisamos modificar no arquivo `config.php`.

database.php

O arquivo `database.php` contém as definições de banco de dados. Se você ainda não criou um banco de dados MySQL para este tutorial, essa é a hora. Com o banco criado, pegue as informações necessárias e atualize esse arquivo de acordo..

```
1 | $db['default']['hostname'] = 'localhost';
2 | $db['default']['username'] = 'root';
3 | $db['default']['password'] = '';
4 | $db['default']['database'] = 'task_board';?
```

Não precisamos alterar as outras configurações.

routes.php

O arquivo `routes.php` é bem poderoso. Ele permite modificar como as URL são tratadas definindo as suas próprias rotas – significa que você pode forçar a URL a funcionar diferente de `controller/method`. Se você modificar muita coisa aqui, pode ficar complicado para outro desenvolvedor descobrir o que sua aplicação está fazendo. No nosso caso a mudança é bem simples.

```
1 | $route['default_controller'] = "login";?
```

Modifique o `default_controller` de `welcome` para `login`. Quando acessarmos `http://localhost/task_board_tutorial`, o CI vai buscar o nome do controlador aqui. Ou seja, estamos enviando o usuário diretamente para o controlador `login` (que criaremos mais tarde).

Agora finalizamos a instalação do CI. Está na hora de ir para o `phpmyadmin` ou sua aplicação MySQL preferida.

Banco de Dados

Nesse ponto precisamos apenas de duas tabelas de banco de dados, `user` e `ci_sessions` (utilizada pelo CI para gerenciar as sessões). Então utilize os seguintes scripts para criar as tabelas no seu banco de dados.

user

```
1 CREATE TABLE IF NOT EXISTS `user` (  
2   `id` int(11) unsigned NOT NULL AUTO_INCREMENT,  
3   `email` varchar(255) NOT NULL,  
4   `password` varchar(40) NOT NULL,  
5   `level` tinyint(4) NOT NULL,  
6   `date_created` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,  
7   PRIMARY KEY (`id`),  
8   UNIQUE KEY `email` (`email`)  
9 ) ENGINE=InnoDB DEFAULT CHARSET=latin1 ;
```

A tabela e usuário terá ID, email, password, level (nível) e a data em que foi criado.

ci_sessions

```
1 CREATE TABLE IF NOT EXISTS `ci_sessions` (  
2   `session_id` varchar(40) NOT NULL DEFAULT '0',  
3   `ip_address` varchar(16) NOT NULL DEFAULT '0',  
4   `user_agent` varchar(120) NOT NULL,  
5   `last_activity` int(10) unsigned NOT NULL DEFAULT '0',  
6   `user_data` text NOT NULL,  
7   PRIMARY KEY (`session_id`),  
8   KEY `last_activity_idx` (`last_activity`)  
9 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Essa tabela é padrão do CI e você pode verificar no guia do usuário (http://codeigniter.com/user_guide/libraries/sessions.html).

Procedimento de Instalação

Agora vamos codificar um pouco. Como nosso banco de dados está vazio, precisamos um procedimento para criar a primeira entrada na tabela de usuários e nos permitir fazer o login. Isso é o que faremos em seguida:

- Criar o modelo (`user_model.php`) dentro do diretório `application/models`.
- Criar o controlador (`install.php`) dentro do diretório `application/controllers`.
- Criar a visão (`login.php`) – vamos utilizar a mesma para o login – dentro do diretório `application/views`.

user_model.php

Crie o arquivo no diretório `application/models` e adicione o seguinte conteúdo:

```
1 <?php  
2 class User_model extends CI_Model {  
3     private $salt = 'r4nd0m';  
4     public $USER_LEVEL_ADMIN = 1;  
5     public $USER_LEVEL_PM = 2;  
6     public $USER_LEVEL_DEV = 3;  
7 }
```

Essa classe vai conter todos os métodos e propriedades do nosso módulo. A primeira propriedade é o `$salt`, que vai ser usado para ajudar a encriptar nossas senhas. Você pode alterar isso para o que vocês quiser... Nós também definimos 3 níveis de usuário utilizando diferentes variáveis. A partir de agora vamos criar os métodos que vamos precisar para o procedimento de instalação. Todos esses métodos devem ir dentro da classe.

```
1 public function get($id = false)  
2 {  
3     if ($id) $this->db->where('id', $id);  
4     $this->db->order_by('email', 'asc');  
5     $get = $this->db->get('user');  
6     if($id) return $get->row_array();  
7     if($get->num_rows > 0) return $get->result_array();  
8     return array();  
9 }
```

O método `get` é responsável por buscar as informações do banco de dados. Como você pode ver, utilizamos aqui a biblioteca de banco de dados do CI (`$this->db`). Para poder utilizá-la, nós precisamos carregar ela primeiro. Porque não estamos fazendo isso neste método? Porque podemos utilizar o `autoload.php` para carregar automaticamente. Então vá para `config/autoload.php` e defina o seguinte:

```
1 $autoload['libraries'] = array('database');
```

Porque fazemos assim? Porque nós vamos utilizar a biblioteca de banco de dados praticamente em todo nosso sistema. Desta maneira a biblioteca sempre estará disponível. A outra opção seria carregá-la utilizando `$this->load->database()` nos locais em que fosse preciso.

Voltando para o método `get`. Estamos utilizando um parâmetro `$id` com um valor padrão de `false`. Isso permite que o método seja utilizado para buscar uma entrada específica (caso passemos um valor para o `$id`) ou várias entradas (caso não passemos nenhum parâmetro). Isso é feito na primeira linha do método, utilizando o um IF e o método `$this->db->where()`, que define a condição WHERE na nossa query.

Depois, definimos o ORDER BY usando o email e executamos a query com o comando `$this->db->get()`. Para retornar os resultados corretamente, utilizamos o `$get->row_array()` caso o `$id` tenha sido definido (o que significa que temos apenas um resultado, então queremos o array somente deste registro). Caso o `$id` não tenha sido definido e o número de resultados for maior que zero, nós retornamos um array com todos os registros, não importan quantos sejam. Se a query não retornar nada, o método retornará um array vazio.

```
1 public function create($data)
2 {
3     $data['password'] = sha1($data['password'].$this->salt);
4     return $this->db->insert('user', $data);
5 }
```

O método `create` insere a informação do usuário no banco de dados. Na primeira linha nós estamos usando a função SHA1 do PHP para encriptar a senha do usuário, concatenada com o “salt” (a primeira propriedade da nossa classe). Depois disso retornamos o resultado da inserção (que será falsa se ocorrer um erro ou o ID do usuário se tudo correr bem).

install.php

Antes de começarmos, não esqueça de remover o arquivo `welcome.php`. Crie o arquivo `install.php` no diretório `application/controllers` e adicione o seguinte conteúdo:

```
1 <?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');
2
3 class Install extends CI_Controller {
4 }
```

Essa é a classe que terá nossos métodos do controlador. Para acessá-la basta entrar em `http://localhost/task_board_tutorial/install`. Nós precisamos de dois métodos dentro dessa classe, um para carregar a visão onde poderemos entrar com a informação e outra para criar o usuário no banco de dados.

```
01 public function index()
02 {
03     $data = array();
04
05     // Check if users are already there
06     $this->load->model('user_model');
07     $users = $this->user_model->get();
08     if($users){
09         $data['already_installed'] = true;
10     } else {
11         $data['already_installed'] = false;
12     }
13
14     // Load View
15     $data['page_title'] = "Installation";
16
17     $data['email'] = '';
18     $data['password'] = '';
19
20     $this->load->view('login', $data);
21 }
```

O método `index()` é chamado por padrão quando você acessa esse controlador. Ele vai verificar se existem usuários no banco de dados para saber se a instalação deve acontecer ou não. Isso é feito utilizando o modelo que criamos anteriormente, utilizando o método `get()`. Com essa informação nós definimos a variável ‘`already_installed`’ que vai ser verificada dentro da nossa visão. Posteriormente nós definimos outras variáveis utilizadas dentro da visão e carregamos a visão utilizando `$this->load->view()`.

```
01 public function run()
02 {
03     // Check if users are already there
04     $this->load->model('user_model');
05     $users = $this->user_model->get();
06     if(!$users){
07         $insert = array(
08             'email' => $this->input->post('email'),
09             'password' => $this->input->post('password'),
10             'level' => $this->user_model->USER_LEVEL_ADMIN
11         );
12         $this->user_model->create($insert);
13     }
14
15     // Load View
16     $data['page_title'] = "Login";
17
18     $data['email'] = '';
19     $data['password'] = '';
20
21     $this->load->view('login', $data);
22 }
```

Este método será executado via post pelo form que está dentro da visão (que vamos criar logo depois). Ele verifica mais uma vez por usuários no banco de dados. Caso o banco esteja vazio (a variável `$users` é falsa), um array com a informação do usuário é criado utilizando a variável `post` (no CI nós utilizamos `$this->input->post()`) e passada como parâmetro para o método `create` do nosso modelo.

A mesma visão/view é carregada mais uma vez, mas dessa vez será aberto o form para o login (parece a mesma coisa, mas as ações são diferentes). Vamos finalizar essa seção com a view.

login.php

Antes de criarmos esse arquivo, apague o arquivo *welcome_message.php*. Não precisaremos dele. Agora continue criando o *login.php* dentro do diretório *application/views*. Esse arquivo é bem simples, por isso vou colar tudo de uma vez.

```
<?php $ass="home-title blue-gradient">Simple Task Board</div>
<?php if(isset($already_installed) && $already_installed) { ?>
    <p>Task Board has already been installed.
</p>
<?php } else { ?>
    <?php if(isset($already_installed) && !$already_installed) { echo form_open('install/run'); }
    else { echo form_open('login/validate'); } ?>

    <?php echo form_label('Email', 'email'); ?>
    <?php echo form_input('email', $email); ?>
    <br/>
    <?php echo form_label('Password', 'password'); ?>
    <?php echo form_password('password', $password); ?>

    <?php if(isset($error) && $error) { ?>
    <p class="error">That's not right! Please check your information and try again.</p>
    <?php } ?>

    <?php if(isset($already_installed) && !$already_installed) { echo form_submit('install', 'Install', 'class="btn-blue"'); }
    else { echo form_submit('login', 'Login', 'class="btn-blue"'); } ?>

    <?php echo form_close(); ?>
</div>
```

Vou explicar apenas as partes principais. Essa visão é apenas um form de login com email e senha. Mas tem algumas validações dentro do arquivo. Primeiro verificamos pela variável *\$already_installed* (lembre que nós definimos essa variável dentro do controlador – método *index()*). Se for true, vai mostrar uma mensagem indicando que o app já foi instalado.

Caso não tenha sido definida, ou tenha sido definida como false, vai mostrar o form utilizando *form_open()*. Mas tem uma diferença entre não ser definida e ser falsa. Você lembra que no método *run()* do controlador nós não definimos essa variável? Isso porque o form será aberto com a ação 'login/validate'. Caso essa variável tenha sido definida como false, o form será aberto com a ação 'install/run'. Essas ações executarão diferentes métodos quando o submit for clicado:

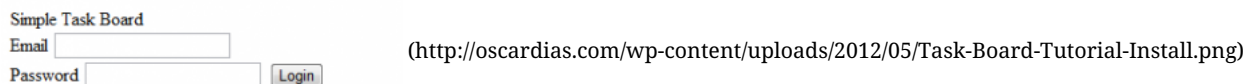
- 'login/validate': executa o método *validate()* dentro do controlador *login* (que será criado na próxima semana)
- 'install/run': executa o método *run()* dentro do controlador *install* (definido anteriormente)

Com essa lógica podemos utilizar a mesma visão para diferentes propósitos, instalação e login. Estou fazendo isso desta maneira porque sou o único trabalhando neste projeto, sendo que assim fica mais simples para mim. Se você estiver trabalhando em um time, com designers que não sejam familiares com codificação, eu recomendo que você separe a visão em duas, removendo a lógica das visões e adicionando no controlador. Isso faz com que seja mais fácil colocar estilo em coisas diferentes e a entender o que está acontecendo na visão.

Antes de testar nosso app nós precisamos de mais uma coisa. Porque utilizamos as funções do form, nós precisamos carregar o “helper” para form. Isso será feito mais uma vez no arquivo de autoloader. Abra *config/autoload.php* e na seção de helper, defina o seguinte:

```
1 | $autoload['helper'] = array('form');
```

Agora, se você for para http://localhost/task_board_tutorial/install, isso é o que você verá:



Se você verificar o código fonte você vai perceber que não temos as tags *head* ou *body*. Isso é porque não colocamos elas na nossa visão. Por que fiz isso? Porque na próxima semana vamos criar uma biblioteca de template bem simples que vai carregar o cabeçalho e o rodapé para nós. Isso mantém as nossas visões mais limpas e vai nos ajudar quando precisarmos atualizar algo como um CSS ou JavaScript, porque precisaremos fazer isso apenas em um lugar.

Conclusão

Nesse post nós vimos um pouco de como isso vai funcionar. O CI é uma ferramenta muito útil e eu espero que você possa entender isso nestes tutoriais. Na próxima semana vamos ver a biblioteca de template e criar a tela de login, que nos levará de volta a visão/view que criamos hoje. Vamos criar um pouco de CSS também.

CodeIgniter (<http://oscardias.com/br/tag/codeigniter/>) | PHP (<http://oscardias.com/br/tag/php/>) | Quadro de Tarefas (<http://oscardias.com/br/tag/quadro-de-tarefas/>)



Este post é parte da série Criando um App usando CodeIgniter (<http://oscardias.com/br/series/criando-um-app-usando-codeigniter/>).

- Criando um App usando CodeIgniter (Parte 1): Introdução (<http://oscardias.com/br/desenvolvimento/php/codeigniter/criando-um-app-usando-codeigniter-parte-1-introducao/>)
- Criando um App usando CodeIgniter (Parte 2): Instalação

- Criando um App usando CodeIgniter (Parte 3): Login (<http://oscardias.com/br/desenvolvimento/php/codeigniter/criando-um-app-usando-codeigniter-parte-3-login/>)
- Criando um App usando CodeIgniter (Parte 4): Usuários (<http://oscardias.com/br/desenvolvimento/php/codeigniter/criando-um-app-usando-codeigniter-parte-4-usuarios/>)
- Criando um App usando CodeIgniter (Parte 5): Projetos (<http://oscardias.com/br/desenvolvimento/php/codeigniter/criando-um-app-usando-codeigniter-parte-5-projetos/>)
- Criando um App usando CodeIgniter (Parte 6): Tarefas (<http://oscardias.com/br/desenvolvimento/php/codeigniter/criando-um-app-usando-codeigniter-parte-6-tarefas/>)
- Criando um App usando CodeIgniter (Parte 7): Validações (<http://oscardias.com/br/desenvolvimento/php/codeigniter/criando-um-app-usando-codeigniter-parte-7-validacoes/>)

Share



(<https://twitter.com/oscardias>)



(<https://www.facebook.com/oscardias>)

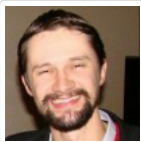


(<http://br.linkedin.com/company/oscardias>)



(<https://plus.google.com/113632041831558412641>)

Sobre Oscar Dias (<http://oscardias.com/br/author/oscardias/>)



Trabalho com desenvolvimento desde 2003 e já utilizei diversas linguagens de programação. Atualmente trabalho com PHP na minha empresa Softerize Sistemas.

< Criando um App usando CodeIgniter (Parte 1): Introdução (<http://oscardias.com/br/desenvolvimento/php/codeigniter/criando-um-app-usando-codeigniter-pa>
Como Agrupar seus Posts WordPress em Séries > (<http://oscardias.com/br/desenvolvimento/php/wordpress/como-agrupar-seus-posts-wordpress-em-series/>)

17 Comments

Oscar Dias - Blog Pessoal

Login

Sort by Best

Share Favorite

Join the discussion...

Brito
22 days ago

Olá amigo, decidi escrever seu tutorial e estou aqui com este problema. Poderia ajudar-me? Obs: estou usando Apache 2.4.7 / Mysql 5.6.15 / php 5.4.24 e CodeIgniter_2.2.0.

Simple Task Board

Fatal error: Call to undefined function form_open() in C:\Program Files (x86)\EasyPHP-DevServer-14.1VC9\data\localweb\application\views\login.php on line 12

1 ^ v Reply

Brito
22 days ago

encontrei o problema!!! É necessário que a linha: \$autoload['helper'] = array('form'); Seja estcrita como: \$autoload['helper'] = array('url','form');

1 ^ v Reply

Eberson
8 months ago

Oscar

Eu deixei fora mas logo depois de postar reli e verifiquei que fui prematuro em postar , mas obrigado pela resposta. Cara muito pratico esse tutorial estava procurando um exemplo para entender a estruturação das paginas e tal agradeço mesmo pela iniciativa de deixar isso a disposição para a gente poder aprender com a tua experiencia.

0 ^ v Reply

Oscar Dias
8 months ago

Que bom que ficou prático... esse era o objetivo. Não tenho atualizado muito o blog, mas estou começando um site chamado Softerize Magazine. Tem outros autores já cadastrados e espero poder fornecer muito mais conteúdo lá - e manter mais atualizado também :)

0 ^ v Reply

Eberson
8 months ago

Boa tarde amigo!

estou seguindo o tutorial porem o apache está me retornando isso no log /var/log/apache2/error.log:
[Wed Jan 29 13:35:14 2014] [error] [client XXX.XX.X.XXX] PHP Parse error: syntax error, unexpected T_PUBLIC in
/home/intranet/www/application/controllers/install.php on line 7

0 ^ v Reply



Oscar Dias → Eberson

8 months ago



Você colocou os métodos dentro da declaração da classe?

```
class Install extends CI_Controller {  
    public function ...  
}
```

Se você declarar uma função fora da classe usando o public você terá esse erro. Pode ser também um "{" ou "}" que esteja sobrando.

0 ^ v Reply



Renato Beletti da Silva

9 months ago



Olá Oscar

Segui o tutorial e tudo esta rodando legal, porem o index.php não consigo retirar... Já habilitei o mod_rewrite no apache, ja edite o config.php e ja criei o .htaccess seguindo o tutorial.

Estou usando Apache2 + PHP 5.5.3

0 ^ v Reply



Oscar Dias → Renato Beletti da Silva

9 months ago



Como estão os seus diretórios e seu arquivo .htaccess? Se o aplicativo for executar em localhost/dir, então esse mesmo dir tem que estar no .htaccess.

0 ^ v Reply



Anderson

10 months ago



Ola estou seguindo se tutorial utilizei o wamp e o xamp mas os mesmos não acessa do endereço direto http://localhost/task_board_tutorial/install sem o index.php o que pode ser a mensagem é de

Not Found

The requested URL /task_board_tutorial/install was not found on this server.

Vlw

0 ^ v Reply



Oscar Dias → Anderson

10 months ago



Tem que habilitar o mod_rewrite do apache. Deve ser esse o problema.

0 ^ v Reply



Eduardo

a year ago



Olá amigo, tive problemas!

Me apareceu este erro:

A PHP Error was encountered

Severity: Notice

Message: Undefined property: Install::\$db

Filename: core/Model.php

Line Number: 51

Fatal error: Call to a member function order_by() on a non-object in

C:\xampp\htdocs\ews\taskboard\application\models\user_model.php on line 10

0 ^ v Reply



Oscar Dias → Eduardo

a year ago



Oi Eduardo,

Deve haver um erro na função get do user_mode. Confere se o código está correto, especialmente o `$this->db->order_by('email', 'asc');`. Caso não seja isso veja se colocou o database no autoload (`$autoload['libraries'] = array('database');`).

Abraço,

Oscar

0 ^ v Reply

 **Cálcio**
2 years ago



Na classe, class User_model extends CI_Model {} e class Install extends CI_Controller {} se ñ me engano é obrigatório o uso do construtor para que se possa fazer o load. Então ficaria assim:

```
User_model extends CI_Model
{
private $salt = 'r4nd0m';
public $USER_LEVEL_ADMIN = 1;
public $USER_LEVEL_PM = 2;
public $USER_LEVEL_DEV = 3;


public function __construct()
{
parent::__construct();
}

...
}
```

e

see more


0 ^ v **Reply**

 **Oscar Dias** → Cálcio
2 years ago



Só é necessário o construtor, com a chamada para o parent::__construct(), se você precisar colocar algo no construtor, como verificar se o usuário está logado, como no controlador project.php por exemplo. Mas eu acho que isso depende também da versão do PHP que você tem. Eu estou com a 5.3.10 e funciona tranquilo.


0 ^ v **Reply**

 **Fernando**
2 years ago



boa tarde amigo, ele não está validando se ja tem usuário no banco e se tento jogar manualmente para o install como vc mostrou ele não encontra.
grande abs e obrigado pelo tutorial, tah show de bola.


0 ^ v **Reply**

 **Oscar Dias** → Fernando
2 years ago



Olá Fernando,
Você conferiu se está com o mod_rewrite do Apache habilitado? Você pode testar se funciona sem o mod_rewrite mudando no config.php:
\$config['index_page'] = 'index.php';
E depois você pode tentar acessar http://localhost/task_board_tutorial/index.php/install. Se funcionar então o problema é o mod_rewrite mesmo...
A validação para verificar se já existe usuário criado é para quando você acessar essa página de install e já existir o usuário.
Abraço

0 ^ v **Reply**

 **Fernando** → Oscar Dias
2 years ago



correto, foi isso mesmo, habilitei o index.php e funcionou,o mod_rewrite não funcionou, obrigado e no aguardo da sequência do tutorial.
grande abs.

0 ^ v **Reply**

 [Subscribe](#)

 [Add Disqus to your site](#)

 [Privacy](#)

