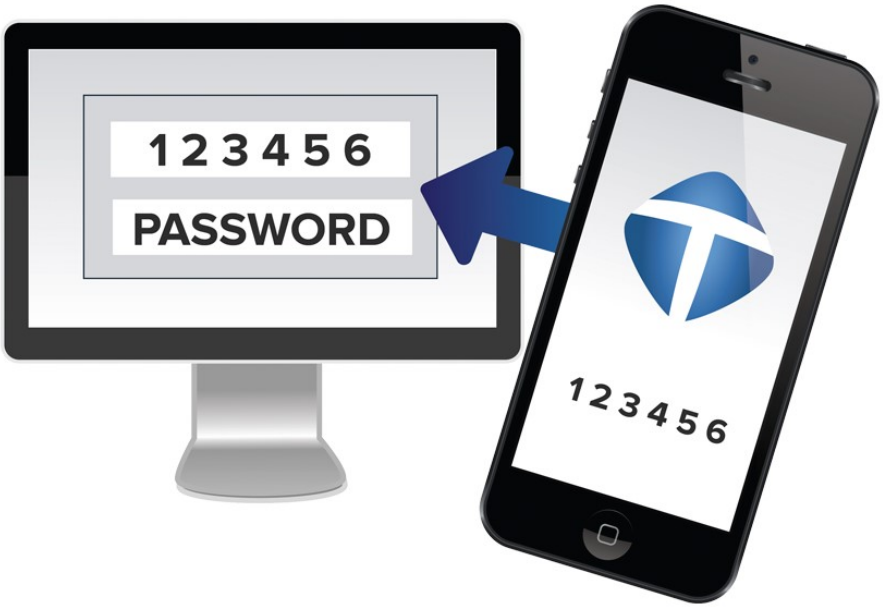
(http://www.universidadecodeigniter.com.br)



Publicado em: 08/04/2016

AUTENTICAÇÃO DE USUÁRIO COM CONTROLE DE ACESSO

Muito comum precisarmos criar aplicações com áreas de acesso restrito, e nessas áreas limitarmos os acessos, de forma a cada usuário ter acesso a um determinado grupo de telas e recursos.

Solucionar esse problema não é algo complicado, com poucas linhas de código é possível criar um sistema de autenticação com controle de acesso, e nas próximas linhas vamos criar uma library para fazer essa autenticação.

Crie um arquivo chamado "Auth.php" dentro do diretório "application/libraries" do seu projeto e coloque o código a seguir nele:

```
1   <?php  if ( ! defined('BASEPATH')) exit('No direct script access allowed');
2   class Auth
3   {
4     private $CI; // Receberá a instância do Codeigniter
5     private $permissaoView = 'sem-permissao'; // Recebe o nome da view correspondente à página informativ
6     private $loginView = 'login'; // Recebe o nome da view correspondente à tela de login
7
8     public function __construct(){
9       /*
10       * Criamos uma instância do CodeIgniter na variável $CI
11       */
12      $this->CI = &get_instance();
13    }
14
15    function CheckAuth($classe,$metodo)
16    {
17
```

```
18      /*
19       * Pesquisa a classe e o método passados como parâmetro em CheckAuth
20       */
21      $array = array('classe' => $classe, 'metodo' => $metodo);
22      $qryMetodos = $this->CI->db->where($array)->get('metodos');
23      $resultMetodos = $qryMetodos->result();
24
25      /*
26       * Caso o método passado ainda não conste na tabela "metodos"
27       * ele é inserido através de $this->CI->db->insert('metodos', $data);
28       */
29      if(count($resultMetodos)==0){
30        $data = array(
31          'classe' => $classe ,
32          'metodo' => $metodo ,
33          'identificacao' => $classe .  '/' . $metodo,
34          'privado' => 1
35        );
36        $this->CI->db->insert('metodos', $data);
37        redirect(base_url($classe . '/' . $metodo), 'refresh');
38      }
39      else{
40        /*
41         * Se o método ja existir na tabela, então recupera se ele é público ou privado
42         * O método sendo público (0), então não verifica o login e libera o acesso
43         * Mas se for privado (1) então é verificado o login e a permissão do usuário
44         */
45        if($resultMetodos[0]->privado==0){
46          return false;
47        }
48        else{
49          $nome = $this->CI->session->userdata('nome');
50          $logged_in = $this->CI->session->userdata('logged');
51          $data = $this->CI->session->userdata('data');
52          $email = $this->CI->session->userdata('email');
53          $id_usuario =  $this->CI->session->userdata('id_usuario');
54
55          $id_metodo = $resultMetodos[0]->id;
56
57          /*
58           * Se o usuário estiver logado faz a verificação da permissão
59           * caso contrário redireciona para uma tela de login
60           */
61          if($nome && $logged && $id_usuario){
62
63            $array = array('id_metodo' => $id_metodo, 'id_usuario' => $id_usuario);
64            $qryPermissoes = $this->CI->db->$this->CI->db->where($array)->get('permissoes');
65            $resultPermissoes = $qryPermissoes->result();
66
67            /*
68             * Se o usuário não tiver a permissão para acessar o método,
```

```
69              * ou seja, não estiver relacionado na tabela "permissoes",
70              * ele deve ser redirecionado para uma tela informando que
71              * não tem permissão de acesso;
72              * caso contrário o acesso é liberado
73              */
74             if(count($resultPermissoes)==0){
75               redirect($this->permissaoView, 'refresh');
76             }
77             else{
78               return true;
79             }
80           }
81           else{
82             redirect($this->loginView, 'refresh');
83           }
84         }
85       }
86     }
87
88   }
```

view raw (https://gist.github.com/jlamim/f912eee9a83e04a605d44e598bdabf37/raw/f6d15dbdae1182815262d82c3a1be0b67c9344c7/Auth.php) Auth.php (https://gist.github.com/jlamim/f912eee9a83e04a605d44e598bdabf37#file-auth-php) hosted with ❤ by **GitHub (https://github.com)**

A classe acima possui um único método, chamado ChechAuth, que fará as seguintes verificações:

- Se o método passado como parâmetro existe na tabela permissões
- Se o método passado é público (0) ou privado (1)
- Se o usuário está logado
- Se o usuário tem permissão para acessar o recurso (método)

Veja a seguir o código SQL para criação das tabelas no banco de dados.

```
1   # Tabela que armazenará os métodos
2   CREATE TABLE `metodos` (
3     `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
4     `classe` varchar(50) DEFAULT NULL,
5     `metodo` varchar(50) DEFAULT NULL,
6     `identificacao` varchar(100) DEFAULT NULL,
7     `privado` tinyint(1) DEFAULT NULL,
8     PRIMARY KEY (`id`)
9   ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
10
11  # Tabela que armazenará as permissões dos cada usuário
12  CREATE TABLE `permissoes` (
13    `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
14    `id_metodo` int(11) DEFAULT NULL,
15    `id_usuario` int(11) DEFAULT NULL,
16    PRIMARY KEY (`id`)
17  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
18
```

```
19  # Tabela que armazenará os usuários
20  CREATE TABLE `usuarios` (
21    `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
22    `nome` varchar(255) DEFAULT NULL,
23    `email` varchar(200) DEFAULT NULL,
24    `senha` varchar(10) DEFAULT NULL,
25    PRIMARY KEY (`id`)
26  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

view raw (https://gist.github.com/jlamim/c43327aba75307314315eaac87616ac0/raw/6e6b07988f59484a141dd9ef50fb0277d3749f6d/database.sql) database.sql (https://gist.github.com/jlamim/c43327aba75307314315eaac87616ac0#file-database-sql) hosted with ❤ by GitHub (https://github.com)

> No repositório no GitHub (link no final do tutorial) tem um arquivo chamado "database.sql", com essa estrutura, além do código da library Auth.

Como utilizar a library criada

Veja a seguir os processos para utilização da library.

Carregamento

Para utilizar a library criada acima você deve carregá-la através do arquivo "application/config/autoload.php" adicionando o seu nome na chave correspondente:

```php
$autoload['libraries'] = array('Auth');
```

Dessa forma, ela estará disponível para uso em qualquer controller e método da aplicação.

Chamada ao método de autenticação CheckAuth

Para chamar o método você utilizará a seguinte linha:

```php
$this->Auth->CheckAuth('nome_classe','nome_metodo');
```

Como parâmetros você deverá passar o nome da classe e do método que o usuário está tentando acessar. O CodeIgniter possui 2 métodos capazes de retornar essa informação, são eles:

```php
//Retorna o nome da classe

$this->router->fetch_class();

//Retorna o nome do metodo

$this->router->fetch_method();
```

Sendo assim, você poderia chamar CheckAuth dessa forma:

```php
$this->Auth->CheckAuth($this->router->fetch_class(), $this->router->fetch_method
```

Essa chamada ao método CheckAuth pode ser realizada em cada um dos métodos, ou então no método __construct() de cada controller da aplicação.

Processo de login

O processo de login da sua aplicação vai armazenar informações do usuário na sessão, e essas informações são recuperadas pela library Auth que foi criada. Verifica no código da library se as informações da sessão que são utilizadas nelas correspondem às informações que você usa na sua aplicação, se forem diferentes, atualize o código da library para que se comporte corretamente.

A parte do código a ser verificada é:

```
$nome = $this->CI->session->userdata('nome');
$logged_in = $this->CI->session->userdata('logged');
$data = $this->CI->session->userdata('data');
$email = $this->CI->session->userdata('email');
$id_usuario =  $this->CI->session->userdata('id_usuario');
```

Rotas e views das telas de login e informação de usuário sem permissão

Não se esqueça de informar no arquivo "application/config/routes.php" as rotas corretas para a exibição das telas de login e usuário sem permissão de acesso, além de criar as respectivas views em "application/views".

Informações importantes

Como a library adiciona automaticamente as classes e métodos na tabela "metodos", caso você altere o nome de algum método no controller, será necessário atualizar também as informações na tabela.

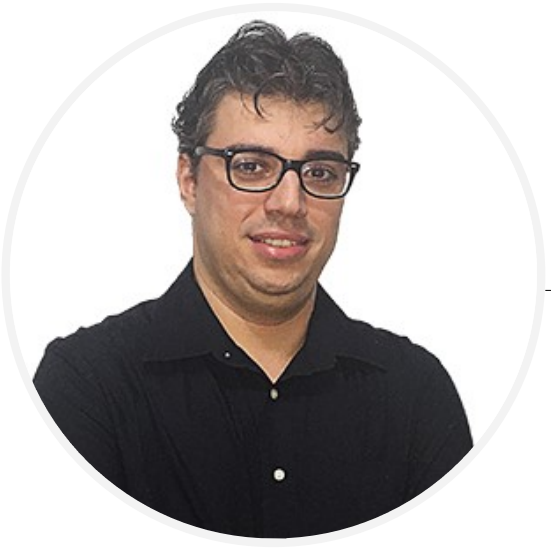
Você pode criar uma interface dentro da sua aplicação para cadastrar e editar manualmente cada um dos métodos adicionados na tabela "metodos".

Tem dúvidas sobre o processo?

Comente com sua dúvida que responderemos o mais breve possível. Ou então envie-nos uma mensagem com a dúvida através da fanpage [https://www.facebook.com/universidadecodeigniter/ [https://www.facebook.com/universidadecodeigniter/]].

[https://github.com/universidadecodeigniter/autenticacao-de-usuario-com-controle-de-acesso]

Tags: autenticação [http://www.universidadecodeigniter.com.br/tag/autenticacao/], biblioteca [http://www.universidadecodeigniter.com.br/tag/biblioteca/], controle de acesso [http://www.universidadecodeigniter.com.br/tag/controle-de-acesso/], permissões [http://www.universidadecodeigniter.com.br/tag/permissoes/]



(http://www.jonathanlamim.com.br)

Jonathan Lamim Antunes

Programador, Escritor, Palestrante e Empreendedor

http://www.jonathanlamim.com.br (http://www.jonathanlamim.com.br)

22 Comments **Universidade CodeIgniter** **João Felipe**

Recommend 2 **Share** **Sort by Best**

Join the discussion…

Leandro Pereira Corso • 10 months ago

Vocês conhecem algum plugin de autenticação que posso instalar via composer?

• Reply • Share ›

Defelper • a year ago

Não entendi a finalidade de se o método não existir no banco, ele cria. Teoricamente eu devo criar todos meus pontos de métodos no banco, correto? Então eu não preciso dessa parte, ou preciso?

• Reply • Share ›

Universidade CodeIgniter **Mod** ➚ Defelper • a year ago

A validação do acesso se dará conforme o cruzamento das informações contidas no banco com a área acessada pelo usuário.

• Reply • Share ›

Defelper ➚ Universidade CodeIgniter • a year ago

Isso eu entendi, minha dúvida foi outra. Porque no primeiro acesso da área acessada pelo usuário tem essa necessidade de criar no banco. Se minha aplicação já possuir todos os métodos no banco, eu posso remover essa inclusão?

• Reply • Share ›

Luis Alves ➚ Defelper • 10 months ago

Só vai criar o método caso o mesmo não exista no banco. Isso é bom porque se você fizer alguma inclusão de método, o sistema já coloca no banco automaticamente pra você ;)

• Reply • Share ›

Universidade CodeIgniter **Mod** ➚ Defelper • a year ago

Até pode, mas se esquecer de lançar algum novo metodo no banco, pode ter problemas no processo de autenticação.

• Reply • Share ›

Livio Rodrigues • a year ago

Tem como criar grupos de acesso, tipo, clientes, empresas, etc... Para que eles acesses determinadas views de acordo com o que é permitido no metodo???

• Reply • Share ›

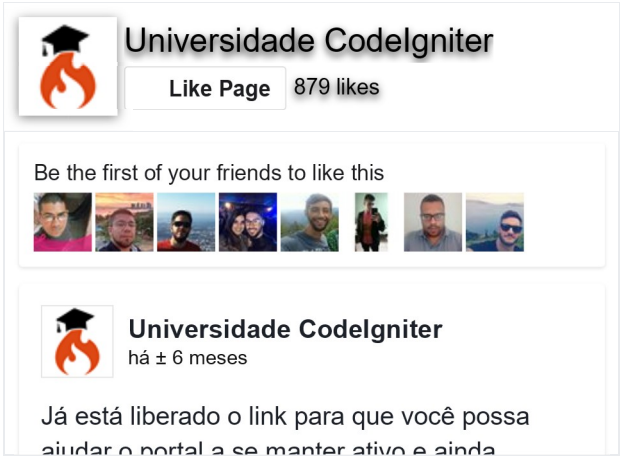
Universidade CodeIgniter **Mod** ➚ Livio Rodrigues • a year ago

Sim, mas aí você vai precisar adaptar essa estrutura, criando as regras de negócio e também as tabelas para armazenar os grupos, quem faz parte de cada grupo e quais as permissões de cada grupo.

1 • Reply • Share ›

roberto joao rosa jr ➚ Livio Rodrigues • 10 months ago

Como eu poderia fazer pra verificar se o usuário pode acessar uma filial por exemplo, eu


Universidade CodeIgniter
Like Page
879 likes
Be the first of your friends to like this
Universidade CodeIgniter
há ± 6 meses
Já está liberado o link para que você possa
ajudar o portal a se manter ativo e ainda