

LAB 12 – Programação em JavaScript, JQuery, e AJAX

Assume-se aqui que já realizou com sucesso o LAB10.

FUNCIONALIDADE REPLY TO POST

1. Adicione a tabela “replies” à sua base de dados

```
a12345@daw:~$mysql -u a12345 -p -h 10.10.23.183 db_a12345
mysql> CREATE TABLE `replies` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `content` text CHARACTER SET utf8 COLLATE utf8_bin,
  `user_id` int(11) DEFAULT NULL,
  `micropost_id` int(11) DEFAULT NULL,
  `created_at` datetime NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT FOREIGN KEY (`user_id`) REFERENCES `users` (`id`),
  CONSTRAINT FOREIGN KEY (`micropost_id`) REFERENCES `microposts` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Nota: se desejar pode realizar o comando utilizando a interface web em <http://all.deei.fct.ualg.pt/phpMyAdmin>

2. Actualize o template `index_template.blade.php` que foi realizado no LAB9, de forma a ter

- o texto “update post” se o utilizador que fez *log in* é o autor do post (já existente)

```
<a href="{{ $post_url }}{{ $blog.id }}">update post</a>
```

- o texto “reply to post” se o post *não pertence* ao utilizador (novidade)

```
<a href="{{ $post_url }}{{ $blog.id }}">reply to post</a>
```

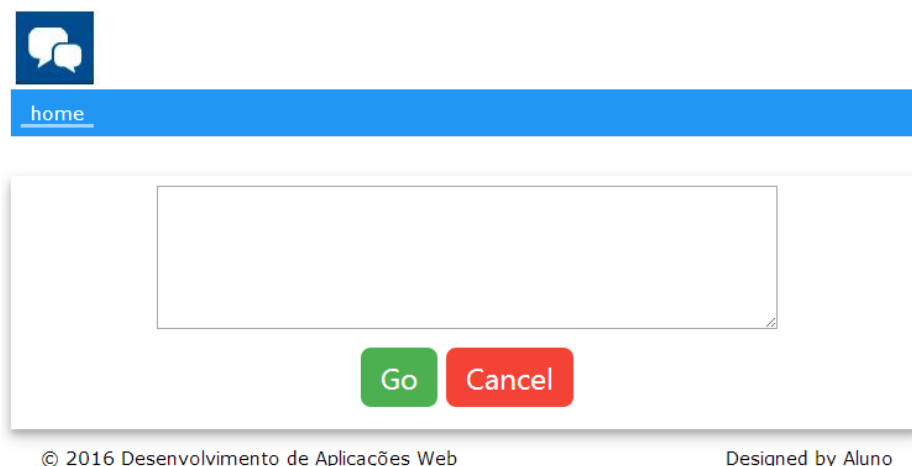


Actualize o método `index()` no ficheiro que define a classe do controlador
(`public_html/LAB9_10/app/Http/Controllers/Blog.php`)

de forma a garantir que em ambos os casos `href` seja o mesmo:

`http://all.deei.fct.ualg.pt/~a12345/LAB9_10/index.php/blog/post/xx`

O template associado ao método “post” é `blog_template.blade.php` (já existente) mostra-se aqui novamente:



Utilize as funções `url()` ou `action()` para garantir que o seu site é portátil.

3. Actualize o método “`post_action`” dentro do ficheiro que define a classe do controlador (`public_html/LAB9_10/app/Http/Controllers/Blog.php`)
Para:

- Chamar o método `Blog_model::new_blog()` caso nenhum `blog_id` seja transmitido no URL (já existente)

- Chamar o método `Blog_model::update_blog($blog_id)` caso o utilizador que fez login seja o autor do post (já existente)
- Chamar o método `Blog_model::new_reply($blog_id)` caso o utilizador que fez login *não seja* o autor do post (**novidade**)

4. Construa o método `new_reply($blog_id)` dentro do ficheiro que define os métodos de acesso à base de dados

(`public_html/LAB9_10/app/Blog_model.php`)

```
public static function new_reply($user_id, $blog_id, $reply)
{
}

```

5. Construa o método `get_replies($id)` em `Blog_model.php`

```
public static function get_replies($id)
{
}

```

Utilize a nova funcionalidade do site para introduzir “replies” em posts de outros utilizadores (em alternativa coloque manualmente algumas “replies” acedendo directamente à base de dados)

6. Actualize novamente o template `index_template.blade.php` e o método `index()` para que em todos os posts em que tenha havido “replies” estas apareçam imediatamente a seguir ao post respectivo:

Esta funcionalidade exige a realização de dois “nested loops”: um loop exterior (“posts”), e um loop interior (“replies”) para receber os dados enviados nos arrays `$blogs` e `$replies`

```
$blogs = Blog_model::get_posts();
for($i=0; $i<count($blogs); $i++)
    $replies[$i] = Blog_model::get_replies($blogs[$i]->id);

```

O resultado deve ser o seguinte screenshot (ou equivalente)*:

* A seccção 10 mostra o código do template `index_template.blade.php`



FUNCIONALIDADE SHOW/HIDE REPLIES COM JAVASCRIPT/JQUERY

7. Atualize novamente o template `index_template.blade.php` e introduza um “botão” no template

```
<button id="b{{$blog->id}}" onclick="myToggle('{{ $blog->id }}')">Show</button>
```

e uma divisão “escondida” que contem as “replies”...

```
<div id="d{{$blog->id}}" style="display:none;">

</div>
```

Utilizando JavaScript ou a livreria JQuery construa a função `myToggle(id)`

```
<script>
function myToggle(id) {
    .
    .
    .
}
</script>
```

que alternando clicar no botão mostra ou esconde as “replies” de um determinado “post”, como se mostra em seguida

- “Replies” escondidas:



- “Replies” visíveis



Teste o funcionamento do botão. Considere esta secção do laboratório concluída quando obtiver a mesma funcionalidade do site

http://all.deei.fct.ualg.pt/~a999993/LV_exame2/blog

FUNCIONALIDADE SHOW/HIDE REPLIES COM AJAX

8. Faça um backup dos ficheiros `Blog.php` e `index_template.blade.php`

9. No método `index()` **remova** a funcionalidade de acesso à tabela de “queries”

```
public function index()
{
    $blogs = Blog_model::get_posts();

    for($i=0; $i<count($blogs); $i++)
```

```
$replies[$i] = Blog_model::get_replies($blogs[$i]->id);
```

~~. . .~~

10. No template index_template.blade.php **remova** o loop interior

```
<div id="d{{$blog->id}}" style="display:none;">
@foreach ($replies[$loop->index] as $reply)

    <div class="w3-card-4">

        <div class="w3-khaki" style="text-align:left;">
            {{$reply->name}} posts: {{$reply->total}}
        </div>

        <div class="w3-container">
            <p>{{$reply->content}}<p>
        </div>

        <div class="w3-khaki" style="text-align:left;">
            {{$reply->created_at}}
        </div>

    </div>

    <br>

@endforeach
</div>
```

11. Construa o método replies(\$id) em Blog.php

```
public function replies($id){
    $data['replies'] = json_encode(Blog_model::get_replies($id));
    echo $data['replies'];
}
```

12. Atualize o ficheiro web.php

```
Route::get('/blog/replies/{id}', 'Blog@replies' );
```

13. Utilizando JavaScript ou a livreria JQuery construa a função myToggle(id)

```
<script>
    function myToggle(id) {
        .
```

```

      .
      .
      .
      }
</script>

```

Esta função é uma função **AJAX** que recebe dadosno formato **JSON**:

Se o botão de um determinado post disser “Show”

- O botão passa a dizer “Hide”
- É feito um GET ao URL `base_url(blog/replies/id)`. Nota: este GET é apenas feito uma vez para um determinado id!
- A `<div>` que se encontra escondida (`style="display:none;"`) passa a ser visível (`style="display:block;"`) e o seu conteúdo é preenchido com os dados da resposta ao GET, formatados com o template que se mostra na secção 10 (ou semelhante)

Se o botão de um determinado post disser “Hide”

- O botão passa a dizer “Show”
- A <div> que se encontra visível (style="display:block;") passa a estar escondida (style="display:none;")

Teste o funcionamento do botão. Considere o laboratório concluído quando obtiver a mesma funcionalidade do site

http://all.deei.fct.ualg.pt/~a999993/LV_exam2/blogJS

http://all.deei.fct.ualg.pt/~a999993/LV_exam2/blogJQ

http://all.deei.fct.ualg.pt/~a999993/LV_exam2/blogNG

REFERÊNCIAS:

- <https://www.w3schools.com/js/default.asp>
- <https://www.w3schools.com/jquery/default.asp>
- https://www.w3schools.com/js/js_ajax_intro.asp
- https://www.w3schools.com/jquery/jquery_ajax_intro.asp

ANEXO 1: Estrutura da base de dados

A estrutura da base de dados pode ser consultada em

<http://all.deei.fct.ualg.pt/phpMyAdmin>

```
CREATE TABLE `users` (  
  `id` int(11) NOT NULL auto_increment,  
  `name` varchar(255) default NULL,  
  `email` varchar(255) default NULL,  
  `created_at` datetime NOT NULL,  
  `updated_at` datetime NOT NULL,  
  `password_digest` varchar(255) default NULL,  
  `remember_digest` varchar(255) default NULL,  
  `admin` tinyint(1) default NULL,  
  `activation_digest` varchar(255) default NULL,  
  `activated` tinyint(1) default NULL,  
  `activated_at` datetime default NULL,  
  `reset_digest` varchar(255) default NULL,  
  `reset_sent_at` datetime default NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `index_users_on_email` (`email`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
CREATE TABLE `microposts` (  
  `id` int(11) NOT NULL auto_increment,  
  `content` text,  
  `user_id` int(11) default NULL,  
  `created_at` datetime NOT NULL,  
  `updated_at` datetime NOT NULL,  
  PRIMARY KEY (`id`),  
  CONSTRAINT FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
CREATE TABLE `replies` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `content` text CHARACTER SET utf8 COLLATE utf8_bin,  
  `user_id` int(11) DEFAULT NULL,  
  `micropost_id` int(11) DEFAULT NULL,  
  `created_at` datetime NOT NULL,  
  PRIMARY KEY (`id`),  
  CONSTRAINT FOREIGN KEY (`user_id`) REFERENCES `users`  
(`id`),  
  CONSTRAINT FOREIGN KEY (`micropost_id`) REFERENCES  
`microposts` (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```