N N Kulobone
Student Number:200995847
Email:nkulobone@hotmail.com

Report on MongoDb

Content

1. Setup and Installation

2. Framework configurations

3. Templates used in mvc

4. Starting up the server

Setup and Installation

MongoDb can be downloaded from the following link:

http://www.mongodb.org/downloads and in my case I downloaded the Win 32-bit which I used with the php drivers starting with the graphic user interface (GUI) for Mongodb which is PHPMoAdmin download at : http://www.phpmoadmin.com/ and the Kohana Framework which is a mvc (mode view control) and can be downloaded at: http://kohanaframework.org/download (version 2.4.0 rc2).

MongoDb was extracted in disk D: on my pc as a mongo folder and I had to create and folder on disk D: (data/db) for storage of MongoDb processes such as databases and collections and all the functionality in the backend [moadmin.php] of it.

I extracted Kohana mvc in the web root which was in wamp\www\ where I also pasted the moadmin.php template so that I could view them in my browser once the connection to mongodb was established using a windows command prompt.

Framework Configuration

- The **View** renders the model into a form suitable for interaction, typically a user interface element.
- The **Controller** is about receiving input and initiating a response by making calls on model objects.

- The **Model** is used to manage information and notify observers when that information changes.

For more on mvc click the link below:

http://dev.kohanaframework.org/wiki/kohana2/Kohana101

Kohana needs to be configured according to the server it is hosted on; in this case it’s the local host wamp server where it is saved in the web root folder wamp\www\ . Copy the routes.php and database.php from the kohana\systems\config file into the kohana\applications\config folder. Open the config.php template found in kohana\applications\config and change the following line:
$config['site_domain'] = 'localhost/kohana/'; because my server is on my machine therefore local host.
Open the routes.php template in the same folder and paste the following code:
<?php defined('SYSPATH') OR die('No direct access allowed.');
$config['_default'] = 'blog'; //because I want the default template to be displayed in the browser [http://localhost/kohana/] to be blog.php which will be created later on.
Open the database.php template which is also found in kohana\applications\config and change the connection settings array to yours, type is mongodb and host is local host for my setup, I downloaded Db.php and Dbo.php which is located in kohana\application\controllers for all the create retrieve update and delete functionality used in the templates I created. Every function [deleting, inserting, updating, viewing] has its own controller class which is extended by that specific application.

Templates used in mvc

I actually created 9 controller templates for the functionality required in the application for the MongoDb blog requirement in the project

- Website.php
  - Contains the 3 links found in the default template [template.php] which are: **Home link** which calls the blog controller found in kohana\application \controllers\blog.php where I required the Db.php for connecting to the database and because in blog.php index function also calls the blog view template which is in kohana\applications\views\pages\blog.php which calls all the blog posts found in mongodb given the following php code:

```
$connection = new Mongo();
$db = $connection->selectDB('nicolas');
```

```
$collection = $db->selectCollection('blog');
$blogs = $collection->find();
```

Upload an article link which calls the upload controller in the controllers' folder which in turn calls the upload template in pages folder which contains the code for uploading a post:

```
$connection = new Mongo();
$db = $connection->selectDB('nicolas');
$collection = $db->selectCollection('blog');
$collection-
>insert(array("author"=>array("username"=>"$uname","firstname"=>"$fname","lastname"=>
"$lname","email"=>"$email","box"=>"$box","zip"=>"$zip","streetname"=>"$street","housenu
mber"=>"$house","town"=>"$town","city"=>"$city"),"post"=>array("title"=>"$title","content"=>
"$content"), "tag"=>array("$tags","$tag1"),"time"=>"$timeStamp" ) );
```

Where every word starting with a $ sign is a variable in php and the data is posted to the same template using the POST method in the form which appears when you click the upload an article link,after filling all the fields and pressing the submit button,the data is posted to the same page and in the code it checks to see if all the variable are not empty in order for it to process the above code.

Search blog by author link which calls the search controller and in turn calls the search template in pages folder and only has one field which is for entering the **author`s user name** for it to process the following code:

```
$connection = new Mongo();
$db = $connection->selectDB('nicolas');
$collection = $db->selectCollection('blog');
$author = (isset($_POST['author']))?strip_tags($_POST['author']):'';
```

The above variables are for establishing the database connection, and the author variable is for the data the user inputs and submits which is sent to the same template again for processing using POST and then the following code is processed:

```
$authorBlogs=$collection->find(array("author"=>array("username"=>$author)));
```

But in my case it did not work therefore I used the following script:

```
$authorBlogs = $collection->find();
```

Find all the posts in blog

```
foreach($authorBlogs as $value)
{
```

```
if($value['author']['username']==$author)
{
Data output
}
}
```

Then check to see from all the posts found if any of the author`s username is similar to the one submitted by the user and if it is output that data.

Delete link uses the blog controller which calls the blog template in pages for deleting the specific post/article because I found it easier to send the data on the same page for this functionality and the code is as follows:

```
$connection = new Mongo();
$db = $connection->selectDB('nicolas');
$collection = $db->selectCollection('blog');
```

Connection establishment to the database nicolas and collection blog.

```
$idG = (isset($_REQUEST['value']))?strip_tags($_REQUEST['value']):'';
```

Request the value of the id of the post/article from the url and then process

```
if((isset($idG)&&(!empty($idG)))&&(isset($delete)&&(!empty($delete))))
{
$collection->remove(array('_id'=>new MongoId($idG)), true);
echo"You have succesfully deleted the post.";
echo"<br /> <a href='blog'>View All Blogs</a>";
}
```

Echo is basically used to output/print data in php.

Add a tag link calls the tags.php controller which in turn calls the tags.php template in pages folder found in the view folder;the Add a tag a href html tag sends the post/article id and once the form is submitted it sends the tag input data and the following code is processed:

```
$connection = new Mongo();
$db = $connection->selectDB('nicolas');
$collection = $db->selectCollection('blog');
$idG = (isset($_REQUEST['value']))?strip_tags($_REQUEST['value']):'';
$submitted = (isset($_REQUEST['sub']))?strip_tags($_REQUEST['sub']):'';
$tag = (isset($_POST['tag']))?strip_tags($_POST['tag']):'';
```

$blog = $collection->findOne(array('_id'=>new MongoId($idG)));
Connects to the database and gets the input data from the form, requests the id of the post/article from the url and we retrieve the specific post using the above findOne function.then the following:

```
$uname = $blog['author']['username'] ;
$fname = $blog['author']['firstname'];
$lname = $blog['author']['lastname'];
$email = $blog['author']['email'];
$box = $blog['author']['box'];
$zip = $blog['author']['zip'];
$street = $blog['author']['streetname'];
$house = $blog['author']['housenumber'];
$town = $blog['author']['town'];
$city = $blog['author']['city'];
$title = $blog['post']['title'];
$content = $blog['post']['content'];
$timeStamp = $blog['time'];
$collection-
>update(array( "author"=>array("username"=>"$uname","firstname"=>"$fname","lastname"
=>"$lname","email"=>"$email","box"=>"$box","zip"=>"$zip","streetname"=>"$street","house
number"=>"$house","town"=>"$town","city"=>"$city"),
"post"=>array("title"=>"$title","content"=>"$content"),
"time"=>"$timeStamp"
),array('$push'=>array("tag"=>"$tag")));
```

Predefined $push function for array is used to add a tag.

Add a comment link calls the addcomments.php controller which in turn calls the addcomments.php template in the view folder under the pages folder: as usual we connect to the database after the user input data is submitted and then use the update function to add a comment to the post/article as follows:

```
$collection->update(array(
"author"=>array("username"=>"$uname","firstname"=>"$fname","lastname"=>"$lname","em
ail"=>"$email","box"=>"$box","zip"=>"$zip","streetname"=>"$street","housenumber"=>"$hou
se","town"=>"$town","city"=>"$city"),
"post"=>array("title"=>"$title","content"=>"$content"),
"tag"=>array("$tags","$tag1"),"time"=>"$timeStamp"
),array('$push'=>array("comments"=>array("name"=>"$username","comment"=>"$contents"
))));
```

Using the $push function inside the update function.

View All Tags link calls the altags.php controller which in turn calls the altags.php viewer template in view\pages. The a href html tag for it passes the post\article id in the url and then we retrieve the data from the url using the following line:

```
$idG = (isset($_REQUEST['value']))?strip_tags($_REQUEST['value']):'';
```

Then find the post/article:

```
$blog = $collection->findOne(array('_id'=>new MongoId($idG)));
```

And extract the data as follows:

```
$num = count($blog['tag']);
$lin = --$num;
$i = 0;
do
{
echo $blog['tag'][$i];
$i++;
if($lin==$i||$lin>$i)
{
echo" | ";
}
}while($i<=$num);
```

View All Comments link calls the alcoms.php controller which in turn calls the alcoms.php template in the view\pages folder, the same logic as view all tags applies here as well where you are taken to a view and shown all the comments after getting the post/article which you clicked for viewing all comments

Starting up the server [MongoBb]

On windows operating system, go to:

Start menu -> click Run and enter cmd -> change to disk d: where I extracted the mongodb and change to the bin folder found inside the mongo folder where the mongodb is saved and run the command **mongod**. Open another command prompt and go to the bin folder and run **mongo.**

Conclusion

The server is actually run first before anything and then work is begun. , First click home link just to have http://localhost/kohana/index.php/blog because if index.php/blog is not there {say you have http://localhost/kohana/index.php}, then all the other links won`t find their respectful pages since i used blog?value=... to pass information.

I will alter it as soon as I am done uploading the Cassandra application.