Load data dynamically in codeigniter

[1]Riddhi Raval,[2]Shrddha Mandaviya,[3]Dipika Chandaliya
[1]Assistant Professor,[2] Assistant Professor,[3] *Software Developer*
[1] Computer Science,
[1] Shree Swaminarayan College Of Computer Science,Bhavnagar,India

***Abstract*— This document gives formatting instructions for authors preparing papers for publication in International Journal of Engineering Research and Development. The authors must follow the instructions given in the document for the papers to be published. You can use this document as both an instruction set and as a template into which you can type your own text. The main intend of this research paper is how to create a simple CRUD(Create, Read, Update,Delete) application with codeigniter. Codeigniter would be the preferred selection when developing bigger projects. Codeigniter study paper you should now have a superior understanding of how Codeigniter framework work.CodeIgniter (CI) is a PHP framework that helps building a full-fledged web application.**

***IndexTerms*— Load data in codeigniter,query of insert/update/delete data in codeigniter**

I. INTRODUCTION:

Databases are the backbone behind any Web application. Without a database, you'd have nowhere to hold all of your data, and SQL queries can become long and cumbersome to type out. Thankfully, CodeIgniter gives us a brilliantly simple way to interact with our Database. The database library also makes changing between database types—from MySQL to Oracle, for example—easier. Loading the Database library is slightly different from loading other libraries. This is because it is large and resides in a different folder, unlike the other libraries.

$this->load->database();

Today we're going to be learning how to connect and then query a database in CodeIgniter. Knowing how to manipulate and read databases is an essential tool for creating any dynamic web application.

We will be using a MySQL database, and for the scope of this tutorial we will not be explaining how to create and populate a MySQL database. If you need help creating a database to test your connection and query on check out the MySQL documentation. You'll also need to know how to create and set up CI routes, controllers, and views.

Automatic Connecting − Automatic connection can be done by using the file application/config/autoload.php. Automatic connection will load the database for each and every page. We just need to add the database library as shown below −

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

Manual Connecting − If you want database connectivity for only some of the pages, then we can go for manual connecting. We can connect to database using this statement

$this->load->database();

CodeIgniter provides a simple way to access the database.It encapsulates the underlying database (MySQL, Oracle, etc)

The connection handling is done by CodeIgniter and configured through:/application/config/database.php Provides security through automatic escaping of inserted data, avoids Cross-Side Scripting (CSS) attacks CodeIgniter uses a modified version of the Active Record Database Pattern.

This pattern allows information to be retrieved inserted, and updated in your database with minimal scripting.

Codeigniter's documentation does offer some insight into how to connect to multiple databases, but I didn't find it satisfactory. It suggests that by passing a second parameter to the loader allows it to return an object which may be then queried like the normal .

$this->db->query();.

II. ACTIVE RECORD: SELECTING DATA:

The Active Record in Codeigniter is quite different to the active record you may find in rails or other frameworks. The way that Active Record in Codeigniter works is that you build up your queries using different functions. For simple queries, you might only need to use one or two functions, but for some you may need to use more-for example, if you have to look for certain conditional, such as where a username and password are the same.

1.$this->db->get():

All of the functions in this section will build SQL SELECT queries. All of the SQL in this section is MySQL; other database systems may differ slightly.

```
$this->db->get();
This would create the SQL query:
SELECT * FROM `table_name`
```

This function has three parameters. The first is the name of the database table. The second lets you set a limit, and the third lets you set an offset.

```
$query = $this->db->get('table_name', 10, 20);
```

This would then produce the SQL query:
SELECT * FROM `table_name` LIMIT 20, 10

2. $this->db->get_where();

This function works in much the same way as the previous function. The only difference is that the second parameter should be passed as an array. The array should have the name of the field and the value to use to fill in the WHERE part of your query.

$query = $this->db->get('table_name', array('id' => $id), 10, 20);
This would produce the following SQL query:
SELECT * FROM 'table_name' WHERE 'id' = $id LIMIT 10, 20

3. $this->db->select();

This function allows you to write the SELECT portion of your query. Take a look at the following example:

$this->db->select('name, username, email');
$query = $this->db->get('users');
The SQL query produced from this function will be:
SELECT name, username, email FROM `users`

You should take note that when using this function, and any of the other functions that let you write a portion of your query, that you still need to use the get() function to actually produce and run the query.

If you are selecting everything from your database (*) then you do not need to use this function as CodeIgniter assumes that you mean to select everything

4. $this->db->group_by()

This function lets you write the GROUP BY portion of your query.

$this->db->group_by('name');
$this->db->group_by(array('name', 'title'));

5. $this->db->order_by()

This lets you write the ORDER BY portion of your query. The first parameter is for your field name. The second is the type of order that you want to use, and can be asc, desc, or random.

$this->db->order_by('name', 'desc');

You can also pass a string as the first parameter.

$this->db->order_by('name desc, title asc');

Multiple function calls can also be used, as for other functions.

$this->db->order_by('name', 'desc');
$this->db->order_by('title', 'asc');

III. ACTIVE RECORD: INSERTING DATA

Inserting data using Active Record is a very simple process, and there are just two functions that you may need to use in order to insert data into your database.

1. $this->db->insert():

This will generate an insert string based upon the data that you supply to it. The first parameter is the name of the table that you want to add the data to, and the second parameter can either be an array or an object of the data.

$data = array('name' => 'Bob Smith', 'email' => 'bob@smith.com');
$this->db->insert('table_name', $data);
This would then produce the following SQL statement:
INSERT INTO mytable (name, email) VALUES ('Bob Smith', 'bob@smith.com')

2. $this->db->set():

This function lets you set data for inserts or updates to your table. This can be used in place of passing an array of data to the insert or update function.

$this->db->set('name', 'Bob Smith');
$this->db->insert('table_name');

If you use multiple function calls they will be properly formatted, depending on whether you are performing an update or an insert.

This function also supports a third parameter. When set to FALSE, this third parameter will prevent data from being escaped.

You can also pass an associative array to this function.

```
$array = array('name' =>'Bob Smith', 'email' => 'bob@smith.com');
$this->db->set($array);
$this->db->insert('table_name');
```

IV. ACTIVE RECORD: UPDATING DATA

Updating data is a highly important part of any web application. CodeIgniter makes this really simple to do. The update function works in largely the same way as the insert function.

1. **$this->db->update();**

This will generate an update string based upon the data that you supply to it. The first parameter is the name of the table you want to add the data to, and the second parameter can either be an **array** or an **object** of the data. The third, optional parameter enables you to set the WHERE clause of your SQL query.

```
$data = array('name' => 'Bob Smith',  'email' => 'bob@smith.com');
$this->db->where('id', 5);
 $this->db->update('table_name', $data);
```

This would then produce the following SQL statement:

```
UPDATE mytable SET name = 'Bob Smith', email = 'bob@smith.com';
```

V. ACTIVE RECORD: DELETING DATA

You can delete data from your tables in a variety of ways. You can either delete fields from a database or empty a database.

1. **$this->db->delete():**

This function accepts two parameters. The first is the name of the table, and the second should be an array from which to build the WHERE clause.

```
$this->db->delete('table_name', array('id' => 5));
```

You can also use the where() function to build the WHERE clause:

```
$this->db->where('id', 5);
$this->db->delete('table_name');
```

An array of table names can be passed into this function, if you wish to delete more than one table.

```
$tables = array('table1', 'table2', 'table3');
$this->db->where('id', '5');
$this->db->delete($tables);
```

2. **$this->db->empty_table():**

This function provides an easy way to delete all of the data from a table. Simply pass the name of the table to the first parameter, to empty it.

```
$this->db->empty_table('table_name');
```

3. **$this->db->truncate():**

This function will generate a TRUNCATE command and run it. It can be used in two ways:

```
$this->db->from('table_name');
$this->db->truncate();
```

or

```
$this->db->truncate('table_name');
```

VI. CODEIGNITER LOGIN FLOWCHART

1 . Create login page, signup page and admin page.
2 Setting up validation to all input field.
3 Check for existing users in database during signup process.
4 Check for username and password in database and show their information stored in database.
5 Create session for admin panel, store users input data in session and destroy session(logout).

We are introducing a flow chart which will give you a clear vision about the objectives of this tutorial.

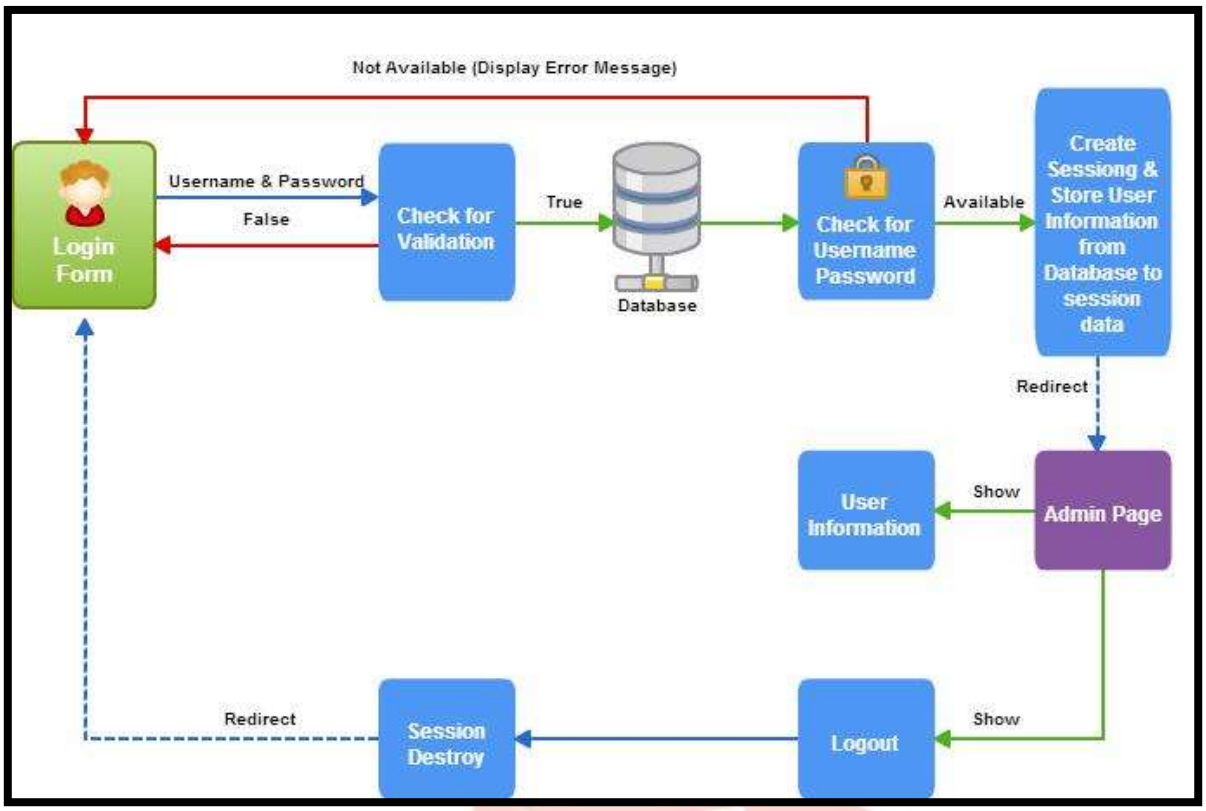

Fig 1: Codeigniter Login Flow Chart

REFERENCES

[1]. Thomas Myer, ”Professional Codeigniter”, Publisher.Wrox(25 July 2008).
[2]. Nitin Reddy Katkam,”Codeigniter v2 Guide, A quick and easy guide to using Codeigniter for PHP developers”, 2013.
[3]. https://www.codeigniter.com/userguide3/installation/index.html
[4]. Eli Orr, Yehuda Zadik, “ Programming with Codeigniter MVC, Build feature-rich web applications using the codeigniter MVC framework”,, Publisher:Packt Publishing ,23th September 2013.
[5]. Packagist. “PHP’s package manager” Packagist.org.
[6]. PHP documentation. “History of PHP” php.net. http://php.net/manual/en/history.php.php
[7]. https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm
[8]. https://en.wikipedia.org/wiki/CodeIgniter
[9]. W3techs. “W3Techs - World Wide Web Technology Surveys”, w3techs.com.