

ICTCore APIs

How ICTCore APIs work

In order to understand that how ICTCore API works in better way we will discuss a complete **CRUD** working functionality of a module of ICTCore. Lets select and discuss the **Contact** module with CRUD functionality.

➤ Create a new Contact

In the **ContactApi** (*wwwroot/core/ApiConatctApi.php*) There is a **Create (function)**method is **ContactAPI** class.

Create method details

- It support the **Post** method
- url will be (<http://ServerIpOrUrl/api/contacts>)

ATTRIBUTES (* required)

- **first_name***
- **last_name***
- **phone***
- **email**
- **address**
- **custom1**
- **custom2**
- **custom3**
- **description**

When a fom post with the above mentioned attribute on the (<http://ServerIpOrUrl/api/contacts>) Url. It will hit the **Create** method and create call the **Contact** module class located (*wwwroot/core/Contact.php*) and pass all above attribute to the **Contact** module.

Contact Module

When **Contact** Module is called, first of all it's **__construct** function is called and set all the posted attribute with it's **fields** defined in **\$fields** public static variable at the top of this class.

This \$this->set() Method

The second process of **ContactApi** set all variables according to database table fields.

The Save Method(function)

The third process of posted form is call the **save()** method. When **\$this->sat()** method is performed, The **save()** will call the **DB::update** class to run the query in mysql database. The **Save** function is define in (wwwroot/core/**Contact.php**) *public function save()*.

The DB class

In ICTCore **DB** class is use for all kind of MYSQL queries. It is located in (wwwroot/core/lib/**DB.php**)

As above mentioned we pass data to **DB::update()**, We will pass variable in this sequence. **DB::update('table_name', \$data_array, \$table_record_id)**

The first parameter is table name of mysql.

The second parameter is array of that data which is posted from form.

The third parameter is table record id. And it is also optional. If it is provided it will update the record and if it is not provided, it will create the new record in Database.

➤ Get a existing/saved Contact

In the **ContactApi** (wwwroot/core/*ApiConatctApi.php*) There is a function named **read()**

When ([http://ServerIpOrUrl/api/contacts/\\$contact_id](http://ServerIpOrUrl/api/contacts/$contact_id)) is called. The **read(\$contact_id)** is execute.

Read method details

- It support the **Get** method
- url will be ([http://ServerIpOrUrl/api/contacts/\\$contact_id](http://ServerIpOrUrl/api/contacts/$contact_id))

ATTRIBUTES (* required)

- **\$contact_id**

When the above mentioned URL is called and **Read** function is triggered with **Contact** module by passing a **\$contact_id** parameter, e.g

```
return new Contact($contact_id);
```

When **ID** is passed to **Contact** module class. It will call the class **__construct** function and in **__construct** function **\$this->load()**; function will be called, In **load** function the **DB::query()**, class is called with ID as third parameter of ID, It will load the contact according to provided contact ID.

➤ Update the existing Contact

In the **ContactApi** ([wwwroot/core/ApiContactApi.php](#)) There is a **Update (function)** method in **ContactAPI** class.

Update method details

- It supports the **PUT** method
- url will be ([http://ServerIpOrUrl/api/contacts/\\$contact_id](#))

ATTRIBUTES (* required)

- **contact_id***
- **first_name***
- **last_name***
- **phone***
- **email**
- **address**
- **custom1**
- **custom2**
- **custom3**
- **description**

When the form is submitted to the mentioned update url with **ID**, The

public function update(\$contact_id, \$data = array()) will be called with the 2 parameters, the first will be Contact ID and second will be all other form fields attribute in array of **\$data** variable. From here the same procedure will be followed

as in create function e.g **Set** method and **Save function** as it describe above in create new contact.

➤ **Delete existing Contact**

In the **ContactApi** (*wwwroot/core/ApiConatctApi.php*) There is a **remove (function)** method is **ContactAPI** class.

Remove method details

- It support the **DELETE** method
- url will be (*http://ServerIpOrUrl/api/contacts/\$contact_id*)

ATTRIBUTES (* required)

- **contact_id***

When the form submitted to the mentioned remove url with **ID**, The

public function remove(\$contact_id) will be called with the parameter Contact ID,

It will triggered with **Contact** module by passing a **\$contact_id** parameter, e.g

```
$oContact = new Contact($contact_id);
```

```
$result = $oContact->delete();
```

When **ID** is passed to **Contact** module class. It will call the class **__construct** function and in **__construct** function **\$this->load();** function will called, In **load** function the **DB::query()**, class is called with ID as third parameter of ID, It will load the contact according to provided contact ID.

After load the contact from database calling **->delete()** instance / method , will delete the loaded contact which was load by providing by ID.

Delete method/function

The **delete** function is placed in **Contact** module, When it is called, It will run a delete query e.g **DB::delete(self::\$table_link, 'contact_id', \$this->contact_id);**

➤ List of all Contacts

In the **ContactApi** (`wwwroot/core/Api/ConatctApi.php`) There is a **list_view** (function) method is **ContactAPI** class.

list_view method details

- It support the **GET** method
- url will be (`http://ServerIpOrUrl/api/contacts`)

When the **list_view** function is called it will call the **search()** method with **array** parameter, By default array parameter is empty and **search** function in **Contact** module will load all contacts.

Search function parameter

If array parater is provided to search function, the search function use it as filter like where condition in mysql. And return the data according to provided parameter/filters.

➤ Create new API

In all files (`wwwroot/core/Api/any_file_here.php`) before defining any method there is a detail about it functionality , url and paramters e.g

```
/**
 * Gets the contact by id
 *
 * @url GET /contacts/$contact_id
 */
```

Everytime when a developer need to create new API in ICTCore he just need to define function / method detail. And he will get the new API url as he define here.

➤ User Authenticate

How login / Authenticate works in ICTCore

In the **AuthenticateApi** (`wwwroot/core/Api/AuthenticateApi.php`) There is a **create (function)** method is **AuthenticateApi** class.

Create a Authentication

When (`http://ServerIpOrUrl/api/authenticate`) is called. The **create()** is execute.

Create / Authenticate method details

- It support the **POST** method
- url will be (`http://ServerIpOrUrl/api/authenticate`)

ATTRIBUTES (* required)

- **username**
- **password**

When the above mentioned URL is called and **create** function is triggered with **User** module module class located (`wwwroot/core/User.php`), When the **create** function is called it will call the **authenticate()** method e.g

User::authenticate(\$credentials, \$key_type)

When **credentials** is passed to **authenticate** function in **User** class. It will check if username or password is correct or not.

Generate Token

When Username and password is match by **authenticate** method it will create the Bearer token e.g

```
$oUser = User::authenticate($credentials, $key_type);
```

```
$oUser->token = $oUser->generate_token();
```

After generate token it will set token expiry date for 1 years. And return the user detail with **Token**, Now this **token** will be use in every api to get perform any task.