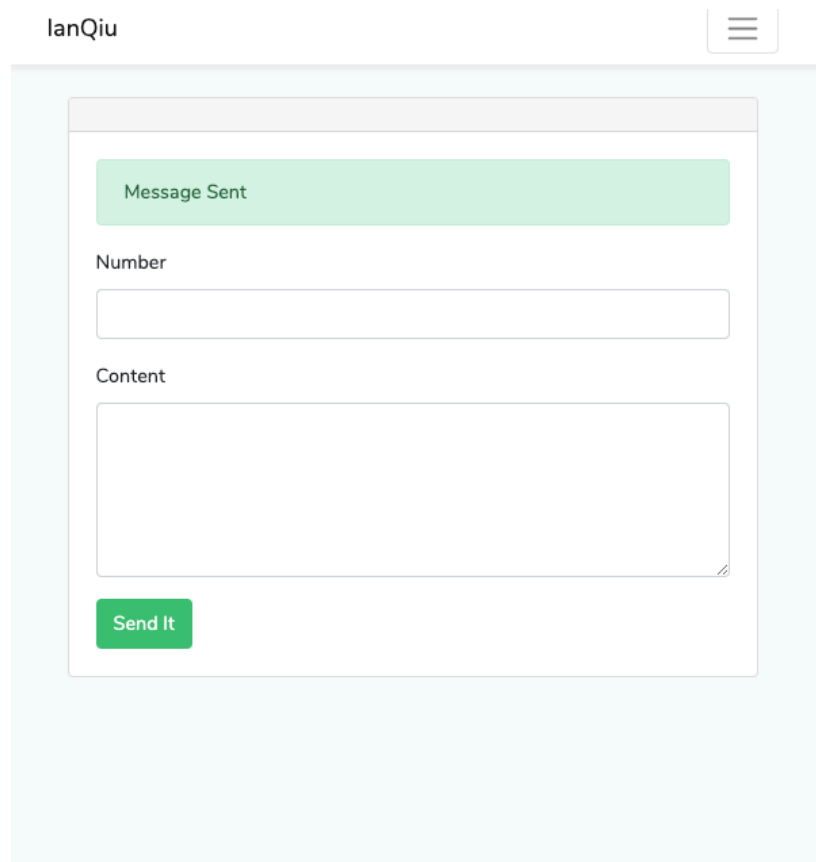


Message sending page:



The screenshot shows a web application interface for sending messages. At the top left, the text "lanQiu" is displayed. At the top right, there is a hamburger menu icon. The main content area is a light blue box containing a white form. The form has a green header bar with the text "Message Sent". Below this, there is a label "Number" followed by a text input field. Underneath the input field is a label "Content" followed by a larger text area. At the bottom of the form is a green button with the text "Send It".

Live:

<http://ec2-13-211-46-235.ap-southeast-2.compute.amazonaws.com/laraapp/public/sms>

Git:

<https://github.com/qiuxan/laraapp>

Files Involved:

1. /composer.json

Adding 'files' in autoload to activate the helpers.php as sending functions are in the helpers file, composer.json and developer needs to run "composer dumpautoload" in command line once to enable the change.

```
"autoload":  
  "files": [
```

```
"bootstrap/helpers.php"  
]
```

2. /bootstrap/helpers.php

This file is holding the sms sending functions . Before developers can actually send sms via alibabacloud sms service (<https://www.alibabacloud.com/product/short-message-service>), we should register an Alicloud account and get the AccessKey ID and Access Key Secret.

Also, developers needs require the dependency to use the AliCloud service: entering “composer require alibabacloud / client” in the command line.

There are 3 values are saved in .env file, and they are ALIYUN_ASSESSKEY, ALIYUN_ASSESSESECRET and APP_NAME. Developers should use their own keys and APP_NAME when sending messages.

3. /app/Http/Controllers/SmsController.php

This is the controller of sending message.

The index function it to let user to see the view.

The store function includes changing the Australian number to the format that can be read by AliCloud Service.

It will check the .env file to make sure ALIYUN_ASSESSKEY and ALIYUN_ASSESSESECRET is not empty.

It will also use message in session to let the user know whether sms is sent or not.

The request sent from the view by method POST is SmsRequest and it will be mentioned in 4.

4. /app/Http/Requests/SmsRequest.php

This is to validate the data before processing to the sendMessage function.
This is to make sure the number is a mobile number and the sms is not empty.

It also used customised error message to make the error message make more sense to the user.

5. /resources/views/sms.blade.php

This is the view of the sms page. Bootstrap is used to make sure it is fully responsive.

`$errors` contains the failed validating messages and it is generated by `SmsRequest`.

`session()->get('success')` contains the the message telling the user that the sms is sent successfully. It is defined in `SmsController` just before redirect to the sms page.

`session()->get('error')` indicates the errors in related to wrong `ALIYUN_ASSESSKEY` or `ALIYUN_ASSESSECRET`. It is defined at the `helpers.php` within the `sendMessage` function.

6. /routes/web.php

The defines the route of the sms sending app. We you use `get` method to trigger the `index` function in `SmsController` so we can view the view and when click 'Send', it will trigger the `store` function.