**Before starting work , run following comands**

1.composer dump

2.composer install

3.composer update

**index.php**

Index.php is genereal file, index.php have spl\_autoload\_register function. Autoload function includes our classes

**class AmazonApiMonitoring**

**public function \_\_construct ($config)**

1. Initialaizing private configuration properties

**@param *array*** $config - contains array with configuration data

**@return** void

**public function run()**

1. This method is main which run index.php

2. Call private method getUrl which returned array with key code (0 or 1)

2.1If code==0 call dp function sending private method SendMail

2.2 Else output success message

3. Call createInstance private method and check returned array with key code (0 or 1)

3.1. 1If code==0 call dp function sending private method SendMail

3.2 Else output success message

4. Call getInstance private method and check returned array with key code (0 or 1)

4.1 If code==0 call dp function sending private method SendMail

4.2 Else output success message

5. Call getSpeed private method and check returned array with key code (0 or 1)

5.1 If code==0 call dp function sending private method SendMail

5.2 Else output success message

6. Call getSpeedByLocation private method and check returned array with key code (0 or 1)

6.1 If code==0 call dp function sending private method SendMail

6.2 Else output success message

7. Call getSpeedDetails private method and check returned array with key code (0 or 1)

7.1 If code==0 call dp function sending private method SendMail

7.2 Else output success message

8. Call displayReport private method and check returned array with key code (0 or 1)

8.1 If code==0 call dp function sending private method SendMail

8.2 Else output success message

**private function getUrl()**

1. Call getUrl private method and check returned array with key code (0 or 1)

2. If returned data column code == 0,

3. Code stops working and calling sendMail private method.

4. Else output success message

**private function createInstance()**

1. Call createInstance private method and check returned array with key code (0 or 1)

2. If returned data column code == 0,

3. code stops working and calling sendMail private method.

4. Else output success message

**private function getInstance($instanceId)**

1. Call getInstance private method and check returned array with key code (0 or 1)

2. If returned data column code == 0,

3. code stops working and calling sendMail private method.

4. Else output success message

**@param *string*** $instanceId - get instance by Id

**@return** array

**private function getSpeed()**

1. Call getSpeed private method and check returned array with key code (0 or 1)

2. If returned data column code == 0,

3. code stops working and calling sendMail private method.

4. Else output success message

**private function getSpeedByLocation($neuStarId)**

1. Call getSpeedByLocation private method and check returned array with key code (0 or 1)

2. If returned data column code == 0,

3. code stops working and calling sendMail private method.

4. Else output success message

**@param *string*** $neuStarId- location id

**@return** array

**private function getSpeedDetails($speedData)**

1. Call getSpeedDetails private method and check returned array with key code (0 or 1)

2. If returned data column code == 0,

3. code stops working and calling sendMail private method.

4. Else output success message

**@param *array* $speedData**- contains array with speed data

**@return** array

**private function displayReport($reportData)**

1.Call displayReport private method and check returned array with key code (0 or 1)

2. If returned data column code == 0,

3. code stops working and calling sendMail private method.

4. Else output success message

**@param *array* $reportData**- contains array with report data

**@return** array

**private function sendMail()**

1. Loging errors using Logger class

2. Send email

**private function isValidResponse($response, $type)**

1.this method checked is valid response json data, if not valid returns (array keys message, code)

**@param *string* $response**- contains response

**@param *string* $type**- contains message type

**@return** array|boolean

**private function curlPostRequst( $url, $postFields )**

1.this method send Curl by URL and Postfields (request data)

**@param *string* $url**- contains url

**@param *string* $postfields**- contains postfields

**@return** array|boolean

**private function postFieldsToStr($postData)**

1.Getting readable construction of the $postData

**@param *array* $postData** - contains respons and requst data

**@return** String

**private function callSuccess()**

1.Calling when response return success

2.Write success message into log file

**@return** void

**private function callError()**

1.Calling when response return error

2.Write error message into log file

**@return** void

**class Helper**

**public static function dataConfiguration($filename = 'locations.ini')**

1.Crating ini file if not exists

2. Checking main\_config.php file existing

3. Checking ini file and get current queue location

4. Udate ini file with current data

5. After geting current location, checking file existing with that name

**@param** ***string*** $filename ini fil name,default name is locations.ini

**@return** ***type***

**public static function write\_php\_ini (array $array, $file)**

1. Create if not exists file

2. Inicializing data by ini file structure

**@param *array*** $array

**@param** ***string*** $file file name

**@return** ***void***

**public static function safefilerewrite($fileName, $dataToSave)**

1. Put data into file using fwrite function

**@param *string*** $fileName file name where puting data

**@param *string*** $dataToSave

**@return *void***

**public static function getQueuStep(array $arr, $key)**

1. Get current location

**@param *array*** $arr contains data

**@param *string*** $key contains key

**@return *string***

**class Logger**

**github link** <https://github.com/Seldaek/monolog/tree/master/src/Monolog>

Monolog sends your logs to files, sockets, inboxes, databases and various web services. See the complete list of handlers below. Special handlers allow you to build advanced logging strategies.

This library implements the PSR-3 interface that you can type-hint against in your own libraries to keep a maximum of interoperability. You can also use it in your applications to make sure you can always use another compatible logger at a later time. As of 1.11.0 Monolog public APIs will also accept PSR-3 log levels. Internally Monolog still uses its own level scheme since it predates PSR-3.