Test framework integration & test cases development

We have a set of API endpoints described as OpenAPI / Swagger specification <http://swagger.io>

This framework generates an SDK code for various languages as well as test cases skeleton.

At the moment we have a full code coverage for PHP language only, but we need to extend this for other languages as well.

List of languages to be covered:

* JAVA
* C#
* JavaScript
* Perl
* C++
* Python

Your goal will be to:

1. Write **SDK initialization code/instructions**
2. **Integrate and setup test framework** for the selected language. I.e. for PHP we using PHPUnit and Composer to solve these needs. Test suite will be run as a part of continuous integration on Linux box.
3. Write a code sample and instructions for the specified language about **retrieving Oauth authorization key**.
4. Develop about **80 test cases** using one of the specified language using existing PHP tests as a reference.

# PHP: SDK initialization code/instructions example

use ProcessMaker\PMIO\ProcessmakerApi;

/\*\* @var ProcessmakerApi $apiInstance \*/

$apiInstance = new ProcessmakerApi;

*Then you should define the server URL ($host), and authorization API key ($key):*

$host = '\_DEFINE\_API\_HOST\_';

$key = '\_DEFINE\_AUTHORIZATION\_KEY\_';

$apiInstance->getApiClient()->getConfig()->setHost("https://$host/api/v1");

$apiInstance->getApiClient()->getConfig()->setAccessToken($key);

*You have option to enable logging and saving debug activities to the file, for example my\_debug.log:*

$apiInstance->getApiClient()->getConfig()->setDebugFile('my\_debug.log');

$apiInstance->getApiClient()->getConfig()->setDebug(true);

# PHP: Test case example

public function testAddUser()

{

try {

$userAtt = new UserAttributes();

$userAtt->setFirstname('Johnny');

$userAtt->setLastname('Doe');

$userAtt->setPassword('password');

$userAtt->setUsername('Username ' . mt\_rand(10000000, 99999999));

$userAtt->setEmail('email@at.com');

/\*\* @var UserItem $result \*/

$result = $this->apiInstance->addUser(new UserCreateItem([

'data' => new User(['attributes' => $userAtt])

]));

$this->assertNotNull($result->getData()->getId());

$this->assertEquals('Johnny', $result->getData()->getAttributes()->getFirstname());

return $result->getData()->getId();

} catch (ApiException $e) {

$this->dumpError($e, \_\_METHOD\_\_);

}

}

# PHP: Getting Oauth authorization key example

Having both client\_id and client\_secret you may retrieve access\_token using *password grant*. Additionally username and password are required to perform the operation.

$args\_for\_bob = [

'grant\_type' => 'password',

'client\_id' => $bobCredentials->getData()->getId(),

'client\_secret' => $bobCredentials->getData()->getAttributes()->getSecret(),

'username' => $bobAttr->getUsername(),

'password' => $bobAttr->getPassword()

];

print\_r(getCredentials($args\_for\_bob, $host));

/\*\*

\* @param array $args Oauth request data

\* @param string $host API HOST

\* @return mixed

\*/

function getCredentials($args, $host)

{

$ch = curl\_init();

curl\_setopt($ch, CURLOPT\_URL, "https://$host/oauth/access\_token");

curl\_setopt($ch, CURLOPT\_POST, 1);

curl\_setopt($ch, CURLOPT\_POSTFIELDS, http\_build\_query($args));

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

$serverResponse = json\_decode(curl\_exec($ch));

curl\_close($ch);

return $serverResponse;

}