Sapi

목록

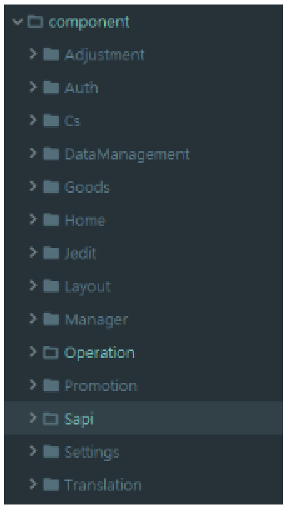
CONCEPT

staff 비지니스 로직들이 restful api 로 제공될 수 있는 하나의 틀

Sapi 라는 명칭으로 하나의 component 를 구성

TOPIC

component



2020-05-06 현재 기준으로 스태프 프로젝트 내에 구분되어져 있는 폴더를 기준으로 component 를 판단

Adjustment	정산 관리
Cs	CS 관리
Translation	번역 관리
Goods	상품
Operation	운영 관리
Promotion	프로모션 관리

object

CS 관리 를 예로 들어서 상점(shop), 사용자(user), 공지사항(notice), 문의(contact), 리뷰(review) 등을 하나의 객체로 구현,

ex) CS 관리 쪽 사용자에 대한 비니지스 로직을 구현할 때, sapi 에서의 component는 cs가 되고 object 는 user 가 됩니다.

object_sequence

object 들이 가지고 있는 unique sequence

sub_object

object 가 가지는 또 다른 object를 sub_object

ex) CS 관리 쪽 사용자가 담당하고 있는 상점

sub_object_sequence

sub_object 들이 가지고 있는 unique sequence

API URL

prefix 는 /tunnel/Sapi/ 입니다.

TOPIC 들을 가지고 api 의 url 을 구성 합니다.

구성

※ object & sub_object 구간은 소문자 / sequence 구간은 숫자만 허용

✓ http://staff.hanpda.com/tunnel/Sapi/(component)/(object)

✓ http://staff.hanpda.com/tunnel/Sapi/(component)/(object)/(object_sequence)

✓ http://staff.hanpda.com/tunnel/Sapi/(component)/(object)/(object_sequence)/(sub_object)

✓ http://staff.hanpda.com/tunnel/Sapi/(component)/(object)/(object_sequence)/(sub_object)/(sub_object_sequence)

GET

✓ ex) 운영 관리 - 쇼핑몰 가이드 정보 목록을 조회

GET http://staff.hanpda.com/tunnel/Sapi/operation/guide

POST

✓ ex) 운영 관리 - 쇼핑몰 가이드 정보를 생성

POST http://staff.hanpda.com/tunnel/Sapi/operation/guide + payload

PUT

✓ ex) 운영 관리 - 쇼핑몰 가이드 정보를 수정

PUT http://staff.hanpda.com/tunnel/Sapi/operation/guide/guide_sequence + payload

DELETE

✓ ex) 운영 관리 - 쇼핑몰 가이드 정보를 삭제

DELETE http://staff.hanpda.com/tunnel/Sapi/operation/guide/guide_sequence

YML 의 구성

Sapi 내 statement.yml

```
www:
  preprocess:
    tunnel:
      membrane:
      - Membrane\AuthMembrane
      - Membrane\CacheMembrane
  tunnel:
    /:
      method: GET|POST|PUT|DELETE|OPTIONS
      kernel: Main
    "/{component:[a-z]+}":
      method: GET|POST|PUT|DELETE|OPTIONS
      kernel: Main
    "/{component:[a-z]+}/{object:[a-z]+}":
      method: GET|POST|PUT|DELETE|OPTIONS
      kernel: Main
    "/{component:[a-z]+}/{object:[a-z]+}/{seq:[0-9]+}":
      method: GET|POST|PUT|DELETE|OPTIONS
      kernel: Main
    "/{component:[a-z]+}/{object:[a-z]+}/{seq:[0-9]+}/{subObject:[a-z]+}":
      method: GET|POST|PUT|DELETE|OPTIONS
      kernel: Main
    "/{component:[a-z]+}/{object:[a-z]+}/{seq:[0-9]+}/{subObject:[a-z]+}/{subSeq:[0-9]+}":
      method: GET|POST|PUT|DELETE|OPTIONS
      kernel: Main
```

Membrane

Auth Membrane

인증 관련

✓ **구현 예정**

Cache Membrane

component + object + QUERY_STRING 이 완전히 동일한 GET 의 요청이라면 이후 해당 component + object 에 변화(POST, PUT, DELETE) 요청이 오기 전까지는 캐싱된 결과를 Response

```php
<?php
namespace Staff\Component\Sapi\Membrane;

use Staff\Component\Sapi\Traits\CacheTrait;
use TS\Http\Proof\Component\ResponseInterface;
use TS\Http\Proof\Route\MembraneInterface;
use TS\Http\Proof\Component\InputInterface;
use TS\Http\Proof\Route\TransporterInterface;
use Staff\Component\Sapi\Traits\ResponseTrait;

class CacheMembrane implements MembraneInterface
{
    use ResponseTrait, CacheTrait;

    /**
     * @param InputInterface $input
     * @param TransporterInterface $transporter
     * @return ResponseInterface
     * @throws \TS\Http\Exception\ComponentException
     */
    public function __invoke(InputInterface $input, TransporterInterface $transporter) : ResponseInterface
    {
        if ($input->getMethod()==='GET') { // 1. requested method  GET
            $cached = $this->getCache($input); // 2. $input QUERY_STRING
            if (strlen($cached)>0) {
                $cached = json_decode($cached, true);
                if (is_array($cached)) {
                    $cached['cached'] = true;
                    return $this->getResponse(200, $cached, 'get'); // 3. Membrane level  response
                }
            }
        }
        return $transporter($input);
    }
}
```

Response

```
{
  "meta":{
    "all_count":0,
    "total_count":1,
    "page":1,
    "limit":15
  },
  "data":[
   {
    "guide_no":76,
    "case_type":"C",
    "case_detail":"",
    "guide_title":"123",
    "guide_content":"<p>123</p>",
    "guide_file_json":"[{"filename":".xlsx","url":"https://staff-pub-dev.cafe24.com/uploads/operation/20200409
/fb1c49aec9a7b19ba9c08eaffb090c35"},{"filename":".pdf","url":"https://staff-pub-dev.cafe24.com/uploads/operation
/20200409/c1faebaac6455bfe9b746cfb1a0f08b0"},{"filename":"2_95_95.png","url":"https://staff-pub-dev.cafe24.com
/uploads/operation/20200409/9985db240cc7c03745fd754d07421880"}]",
    "upd_user_id":"ymbae",
    "upd_user_name":"",
    "upd_datetime":"2020-04-09 13:29:04",
    "res_datetime":"2020-04-09 13:29:04",
    "res_user_id":"ymbae",
    "res_user_name":""
    }
  ],
  "cached":true // <<--  response
}
```

CORS (Cross-Origin Resource Sharing)

> **보통 staff-web front 쪽에서 staff-api server 로의 XML HTTP Request 요청을 하게 될 텐데 이는 결국 교차 출처 리소스 공유 설정을 필요로 하게 됩니다.**

Server

sapi 쪽에서 response 처리를 할 경우 최상단 ResponseTrait 을 사용하게 되고 이쪽에서 response header 설정을 합니다.

```
<?php
namespace Staff\Component\Sapi\Traits;

use TS\Http\Component\Response;
use TS\Http\Proof\Component\ResponseInterface;
use TS\Resource\Stream;

/**
 * Trait ResponseTrait
 * @package Staff\Component\Sapi\Traits
 */
trait ResponseTrait
{
    /**
     * @param int $status
     * @param array $payload
     * @param string $method
     * @return ResponseInterface
     * @throws \TS\Http\Exception\ComponentException
     */
    public function getResponse(int $status = 400, array $payload = [], string $method = 'get') :
ResponseInterface
    {
        return (new Response($status))
            ->withBody(json_encode($this->getSpec($status, $method, $payload)))
            ->withHeader('Content-Type', 'application/json')
            ->withHeader('Access-Control-Allow-Origin', '*')
            ->withHeader(
                'Access-Control-Allow-Headers',
                'Content-Type, Authorization, Content-Length, X-Requested-With'
            )
            ->withHeader('Access-Control-Allow-Methods', 'GET,POST,PUT,DELETE,OPTIONS')
            ->withHeader('Access-Control-Max-Age', 3600);
    }

    /**
     * @param int $status
     * @param string $method
     * @param array $payload
     * @return array
     */
    private function getSpec(int $status, string $method, array $payload) : array
    {
        if ($status>201) {
            return [
                'errors' => $payload
            ];
        } else {
            return $payload;
        }
    }

    /**
     * @param string $fileName
     * @param string $url
     * @return ResponseInterface
     * @throws \TS\Http\Exception\ComponentException
     * @throws \TS\Resource\Exception\StreamException
     */
    public function getResponseDownload(string $fileName, string $url) : ResponseInterface
    {
        $response = new Response();
        $response = $response->withHeader('Content-Type', 'application/octet-stream');
        $response = $response->withHeader('Content-Disposition', 'attachment; filename='.$fileName);
        $response = $response->withBody(new Stream($url));

        return $response;
    }
}
```

추가적으로 http method OPTIONS 의 요청시 세팅된 header 정보만 필요하므로

Main 로직 쪽에 예외 처리가 되어 있습니다.

```
if ($input->getMethod()==='OPTIONS') {
    return $this->getResponse(200, ['message' => '']);
}
```

Front

XML HTTP Request 를 하는 경우 same origin, cross origin 을 판단하여

cross origin 일 때 그에 맞는 request header 구성을 위한 http method OPTIONS 으로 header 조회를 먼저 하는 등의 처리가 필요한데

현재 사용중인 jquery 1.4.1 버전의 .ajax에선 해당의 기능이 없고, .6 버전 이후로 crossDomain 이라는 option 이 있는 것으로 확인되나,

jquery 버전을 올리기에는 퍼블리싱 버전의 js들(table fixed) 등과의 충돌이 예상 되므로,

관련 모듈 원탑인 axios 를 사용 (자동으로 처리 됨)

```
axios.get(
    `${S_API_DOMAIN}/tunnel/Sapi/operation/shop`, {
        params : {
            package: shopSeqNo
        }
    }
).then(({data})=>{
    if (typeof data==='object' && typeof data[0]==='object' && typeof data[0].cnt==='number' && data[0].cnt>0) {
        if (typeof data[0].package_no === 'number') {
            if (confirm(__('    . \r\n    ?'))) {
                document.location.href = '/browse/Operation/shopPackage/create?package_no='+data[0].package_no;
            }
        } else {
            STF.notify(__('  .'));
            packageJS.setReferInfo(data[0].mall_id);
        }
    } else {
        STF.notify(__('   .'));
    }
}).catch(error => {
    console.log('error', error);
});
```

Test

하나의 function 당 하나의 test 를 구성

각 function 에서의 예외 처리, 필수 처리 등 실패들은 code test

실제 등록/수정/삭제는 behavior test

Code test

```
<?php
/**
 * Created by PhpStorm.
 * User:
 * Date: 2020-03-11
 * Time:  2:36
 */
namespace Staff\Test\Sapi\Operation;

use Staff\Bridge\Test\Component\CsTestCase;
use Staff\Component\Sapi\Operation\Package;
use TS\Http\Component\Input;

class PackageTest extends CsTestCase
{
    private $arguments;
    private $spec = [
        'name' => 'Staff\Component\Sapi\Operation\Package',
        'parentName' => '',
        'endLine' => 309,
        'cloneable' => true,
        'properties' => [
            'input',
            'packageModel',
            'status',
            'message',
            'packageNo',
            'logger',
            'phpInput',
        ],
    ];

    public function setUp() : void
    {
        parent::setUp();
        $this->arguments = [
            $this->createMock(Input::class)
        ];
    }

    /**
     * @throws \ReflectionException
     * @throws \TS\Dao\Exception\InvalidParametersException
     */
    public function testSpec() : void
    {
        $this->commonSpecTest($this->spec, new \ReflectionClass($this->getMain()));
    }

    /**
     * @return Package
     * @throws \TS\Dao\Exception\InvalidParametersException
     */
    private function getMain() : Package
    {
        return new Package(...$this->arguments);
    }

    /**
     * @return array
     */
    public function dataGet() : array
    {
        return [
            [
                [], 200, ''
            ],
            [
                [
                    'page' => 1
                ], 200, ''
            ],
            [
                [
                    'type' => 'STAFF'
                ],
                400,
                '{"errors":["type must be in the defined set of values"]}'
            ],
            [
                [
                    'page' => 'one'
                ],
                400,
                '{"errors":["page must be an integer"]}'
            ],
            [
                [
                    'start_date' => '9999-99-99 99:99:99'
                ],
                400,
                '{"errors":["start date must be a valid date"]}'
            ],
            [
                [
                    'guide_name' => 'guide_name',
                    'contract_type' => 'L',
                    'package_type' => 'A',
                    'page' => 1,
                    'limit' => 1,
                    'include_all_count' => 'Y',
                    'sort_key' => 'mall_id',
                    'sort_val' => 'desc',
                    'date_type' => 'res_datetime',
                    'start_date' => date('Y-m-d'),
                    'end_date' => date('Y-m-d'),
                    'keyword_type' => 'res_user',
                    'keyword' => 'staff'
                ],
                200,
                ''
            ],
        ];
    }

    /**
     * @dataProvider dataGet
     * @param array $requestInput
     * @param int $expectedStatusCode
     * @param string $expected
     * @throws \ReflectionException
     * @throws \TS\Dao\Exception\InvalidParametersException
     */
    public function testGet(
        array $requestInput,
        int $expectedStatusCode,
        string $expected
    ): void {
        $input = $this->createMock(Input::class);
        $input->method('getQueries')->willReturn($requestInput);
        $this->arguments = [$input];

        $o = $this->getMain();
        $r = new \ReflectionClass($o);
        $c = $r->getMethod('get')->getClosure($o);
        $response = $c();
        $this->assertEquals($expectedStatusCode, $response->getStatusCode());
        if ($expectedStatusCode!==200) {
            $this->assertEquals($expected, $response->getBody()->getContents());
        }
    }

    public function dataPost(): array
    {
        return [
            [
                [
                    [
                        'staff' => 'test'
                    ]
                ],
                400,
                '{"errors":["type must be provided, but does not exist","guide_name must be provided, '.
                'but does not exist","contract_type must be provided, but does not exist"]}'
            ],
            [
                [
                    [
                        'type' => 'C',
                        'guide_name' => 'shit'
                    ]
                ],
                400,
                '{"errors":["contract_type must be provided, but does not exist"]}'
            ],
            [
                [
                    [
                        'type' => 'C',
                        'guide_name' => 'shit',
                        'contract_type' => 'LOL'
                    ]
                ],
                400,
                '{"errors":["contract type must be in the defined set of values"]}'
            ],
            [
                [
                    [
                        'type' => 'C',
                        'guide_name' => 'staff test',
                        'contract_type' => 'L'
                    ],
                    true
                ],
                400,
                '{"errors":[{"type":"C","guide_name":"staff test","contract_type":"L"}]}'
            ],
            [
                [
                    [
                        'type' => 'C',
                        'guide_name' => 'staff test',
                        'contract_type' => 'L',
                        'res_user_id' => 'say',
                        'res_user_name' => 'hello'
                    ],
                    true
                ],
                400,
                '{"errors":[{"type":"C","guide_name":"staff test","contract_type":"L",'.
                '"res_user_id":"say","res_user_name":"hello","upd_user_id":"say",'.
                '"upd_user_name":"hello"}]}'
            ],
            [
                [
                    [
                        'type' => 'C',
                        'guide_name' => 'staff test',
                        'contract_type' => 'L',
                        'shop_seq_no' => 'hahaha'
                    ],
                    true
                ],
                400,
                '{"errors":["shop seq no must be an integer"]}'
            ],
            [
                [
                    [
                        'type' => 'C',
                        'guide_name' => 'staff test',
                        'contract_type' => 'L',
                        'shop_seq_no' => 55447788
                    ],
                    true
                ],
                400,
                '{"errors":[{"type":"C","guide_name":"staff test","contract_type":"L"}]}'
            ],
        ];
    }

    /**
```

```php
     * @dataProvider dataPost
     * @param array $requestInput
     * @param int $expectedStatusCode
     * @param string $expected
     * @throws \ReflectionException
     * @throws \TS\Dao\Exception\InvalidParametersException
     */
    public function testPost(
        array $requestInput,
        int $expectedStatusCode,
        string $expected
    ): void {
        $input = $this->createMock(Input::class);
        $input->method('getRequests')->willReturn($requestInput[0]);
        $this->arguments = [$input];

        $o = $this->getMain();
        $r = new \ReflectionClass($o);

        if (count($requestInput)===2) {
            $p = $r->getProperty('packageModel');
            $p->setAccessible(true);
            $p->setValue($o, new class {
                public function insert($a)
                {
                    return [
                        'result' => false,
                        'message' => $a
                    ];
                }
            });
        }

        $c = $r->getMethod('post')->getClosure($o);

        $response = $c();
        $this->assertEquals($expectedStatusCode, $response->getStatusCode());
        if ($expectedStatusCode!==200) {
            $this->assertEquals($expected, $response->getBody()->getContents());
        }
    }

    public function dataPut(): array
    {
        return [
            [
                [],
                400,
                '{"errors":["It must have package_no."]}'
            ],
            [
                [''],
                400,
                '{"errors":["It must have package_no."]}'
            ],
            [
                [0],
                400,
                '{"errors":["It package_no does not exist in staff database."]}'
            ],
            [
                [
                    0,
                    new class {
                        public function getOne()
                        {
                            return [
                                'result' => false,
                                'message' => 'query error get one'
                            ];
                        }
                    }
                ],
                400,
                '{"errors":["query error get one"]}'
            ],
            [
                [
                    0,
                    new class {
                        public function getOne()
                        {
                            return [
                                'result' => true,
                                'message' => [0]
                            ];
                        }
                    }
                ],
                400,
                '{"errors":["Request parameter is invalid."]}'
            ]
        ];
    }

    /**
     * @dataProvider dataPut
     * @param array $requestInput
     * @param int $expectedStatusCode
     * @param string $expected
     * @throws \ReflectionException
     * @throws \TS\Dao\Exception\InvalidParametersException
     */
    public function testPut(
        array $requestInput,
        int $expectedStatusCode,
        string $expected
    ): void {
        if (count($requestInput)>0) {
            $input = $this->createMock(Input::class);
            $input->method('getAttribute')->willReturn($requestInput[0]);
            $this->arguments = [$input];
        }
        $o = $this->getMain();
        $r = new \ReflectionClass($o);
        $c = $r->getMethod('put')->getClosure($o);

        if (count($requestInput)===2) {
            $p = $r->getProperty('packageModel');
            $p->setAccessible(true);
            $p->setValue($o, $requestInput[1]);
        }

        $response = $c();
        $this->assertEquals($expectedStatusCode, $response->getStatusCode());
        if ($expectedStatusCode!==200) {
            $this->assertEquals($expected, $response->getBody()->getContents());
        }
    }

    public function dataDelete(): array
    {
        return [
            [
                [],
                400,
                '{"errors":["package_no must be provided, but does not exist"]}'
            ],
            [
                [
                    [
                        'package_no' => 'NO'
                    ]
                ],
                400,
                '{"errors":["There is not able to delete package data."]}'
            ],
            [
                [
                    [
                        'package_no' => 'YES'
                    ],
                    new class {
                        public function getALL($parameters)
                        {
                            return [
                                'result' => true,
                                'message' => [
                                    'meta' => [
                                        'total_count' => 1
                                    ],
                                    'data' => []
                                ]
                            ];
                        }
                    }
                ],
                400,
                '{"errors":["There is not deleted package data."]}'
            ],
        ];
    }

    /**
     * @dataProvider dataDelete
     * @param array $requestInput
     * @param int $expectedStatusCode
     * @param string $expected
     * @throws \ReflectionException
     * @throws \TS\Dao\Exception\InvalidParametersException
     */
    public function testDelete(
        array $requestInput,
        int $expectedStatusCode,
        string $expected
    ): void {
        if (count($requestInput)>0) {
            $input = $this->createMock(Input::class);
            $input->method('getQueries')->willReturn($requestInput[0]);
            $this->arguments = [$input];
        }
        $o = $this->getMain();
        $r = new \ReflectionClass($o);
        $c = $r->getMethod('delete')->getClosure($o);

        if (count($requestInput)===2) {
            $p = $r->getProperty('packageModel');
            $p->setAccessible(true);
            $p->setValue($o, $requestInput[1]);
        }

        $response = $c();
        $this->assertEquals($expectedStatusCode, $response->getStatusCode());
        if ($expectedStatusCode!==200) {
            $this->assertEquals($expected, $response->getBody()->getContents());
        }
    }
}
```

Behavior test

```
<?php
/**
 * Created by PhpStorm.
 * User:
 * Date: 2020-04-01
 * Time:  10:51
 */
namespace Staff\Test\Sapi\Operation\Behavior;

use PHPUnit\Framework\TestCase;
use Staff\Component\Sapi\Operation\Package;
use TS\Http\Component\Input;

/**
 *   /  /  /    .
 *     ,
 * Class BehaviorPackageTest
 * @package Staff\Test\Sapi\Operation\Behavior
 */
class BehaviorPackageTest extends TestCase
{
    /**
     * @throws \ReflectionException
     * @throws \TS\Dao\Exception\InvalidParametersException
     */
    public function testALL() : void
    {
        $guideName = 'behavior test : '.date('YmdHis');

        $input = $this->createMock(Input::class);
        $input->method('getRequests')->willReturn([
            'type' => 'C',
            'guide_name' => $guideName,
            'contract_type' => 'L'
        ]);

        $o = new Package($input);
        $r = new \ReflectionClass($o);

        $c = $r->getMethod('post')->getClosure($o);
        $response = $c();

        #
        $this->assertEquals(201, $response->getStatusCode());

        $input = $this->createMock(Input::class);
        $input->method('getQueries')->willReturn([
            'guide_name' => $guideName
        ]);

        $o = new Package($input);
        $r = new \ReflectionClass($o);

        $c = $r->getMethod('get')->getClosure($o);
        $response = $c();

        #
        $this->assertEquals(200, $response->getStatusCode());

        $body = json_decode($response->getBody()->getContents(), true);

        if (is_array($body) && isset($body['data']) && isset($body['data'][0])) {
            $packageNo = $body['data'][0]['package_no'] ?? 0;

            $input = $this->createMock(Input::class);
            $input->method('getAttribute')->willReturn($packageNo);

            $o = new Package($input);
            $r = new \ReflectionClass($o);

            $p = $r->getProperty('phpInput');
            $p->setAccessible(true);
            $p->setValue($o, '{"type": "S"}');

            $c = $r->getMethod('put')->getClosure($o);

            $response = $c();

            #
            $this->assertEquals(200, $response->getStatusCode());

            $input = $this->createMock(Input::class);
            $input->method('getQueries')->willReturn([
                'package_no' => $packageNo
            ]);

            $o = new Package($input);
            $r = new \ReflectionClass($o);

            $c = $r->getMethod('delete')->getClosure($o);

            $response = $c();

            #
            $this->assertEquals(200, $response->getStatusCode());
        } else {
            #
            $this->assertTrue(false);
        }
    }
}
```