제 9회 한국 커뮤니티 데이

KCD 2020

"커뮤니티의, 커뮤니티에 의한, 커뮤니티를 위한
대한민국 최대 규모의 커뮤니티 소통의 장"

주최 커뮤니티

Codeigniter 한국사용자포럼 · MongoDB Korea · OKKY · TAEYO.NET ASP & ASP.NET · K KOREA · k KOREA · C++ · ASP.NET · Azure · ubuntu · {SLiPP} · DATA BREAK · OSS개발자포럼 · 자바카페. · 한국 Azure 사용자그룹 · 30 · JBUG Korea · KR · TS · DevOps · BACKEND. · 모닝클래스 · WOMEN WHO CODE SEOUL · fast.ai · 한국리눅스커널 개발자모임 · TIZEN · IT Infra Engineer · tidyverse

| 제 9회 한국 커뮤니티 데이 온라인 | 2020년 11월 7일(토) |

nipa 정보통신산업진흥원 National IT Industry Promotion Agency · Microsoft · 미노베이션아카데미 · elastic · LINE · MySQL · CUBRID · eBrain · boneis CONVERGENCE · Festa! · 한빛미디어 Hanbit Media, Inc. · Jpub 제이펍 · 志&禪 지앤선 · 길벗

Korea 제 9회 한국 커뮤니티 데이
Community Day 2020

CodeIgniter4 RESTful API 구현

코드이그나이터 한국 사용자 포럼 한대승(불의회상)

발표자 소개

- (주)포비즈코리아
  프로젝트사업본부 개발 리더
- Codeigniter 한국 사용자 포럼 운영진
- Framework Designer
- PHP, JAVASCRIP

Index

- Codeigniter4
- RESTful API
- Codeigniter4로 RESTful API 만들기
- Q & A

Codeigniter4

Codeigniter

- PHP로 작성된 웹프레임워크
- 2006년 2월 28일 EllisLab에서 첫 공개 버전 발표
- 2014년 10년 6일 브리티시콜럼비아기술대학 관리 및 개발
- 2020년 2월 24일 버전4 발표

Codeigniter의 특징

- 설치가 쉽다.
- 배우기 쉽다.
- 도움 받기 쉽다.
- 태클 걸지 않는다.(MIT 라이센스)

RESTful API

REST(Representational State Transfer)

월드 와이드 웹(WWW)과 같은 분산
하이퍼미디어 시스템을 위한 소프트웨어
아키텍처 한 형식

REST 역사

로이 필딩(Roy Fielding)이 2000년 UC 어바인에서 "Architectural Styles and the Design of Network-based Software Architectures"라는 제목의 2000년 박사 학위 논문에 REST 정의

RESTful API란?

로이 필딩(Roy Fielding)이 정의한

REST 원리를 따르는 API 시스템

RESTful API 목표

- 구성 요소 상호작용의 규모 확장성
- 인터페이스의 범용성
- 구성 요소의 독립적인 배포
- 중간적 구성요소를 이용한 응답 지연 감소, 보안 강화, 레거시 시스템의 인캡슐레이션

RESTful API 조건

- 클라이언트/서버 구조
- 무상태(Stateless)
- 캐시 처리 가능(Cacheable)
- 계층화(Layered System)
- 인터페이스 일관성

RESTful API 활용

- 리소스(자원)의 식별
- 메시지를 통한 리소스의 조작

RESTful API 메소드

- POST : 자원 생성(Create)
- GET : 자원 조회(Read)
- PUT : 자원 수정(Update)
- DELETE : 자원 삭제(Delete)

Codeigniter4로 RESTful API 만들기

준비물

- PHP 7.2 이상 (7.4.X 권장)
- Composer (https://getcomposer.org)
- RESTful API Client

Codeigniter4 설치

- Linux, Mac

  $ composer create-project codeigniter4/appstarter project

- Windows

  c:\> composer create-project codeigniter4/appstarter project

Codeigniter4 실행

- Linux, Mac

  $ cd project

  $ php spark serve

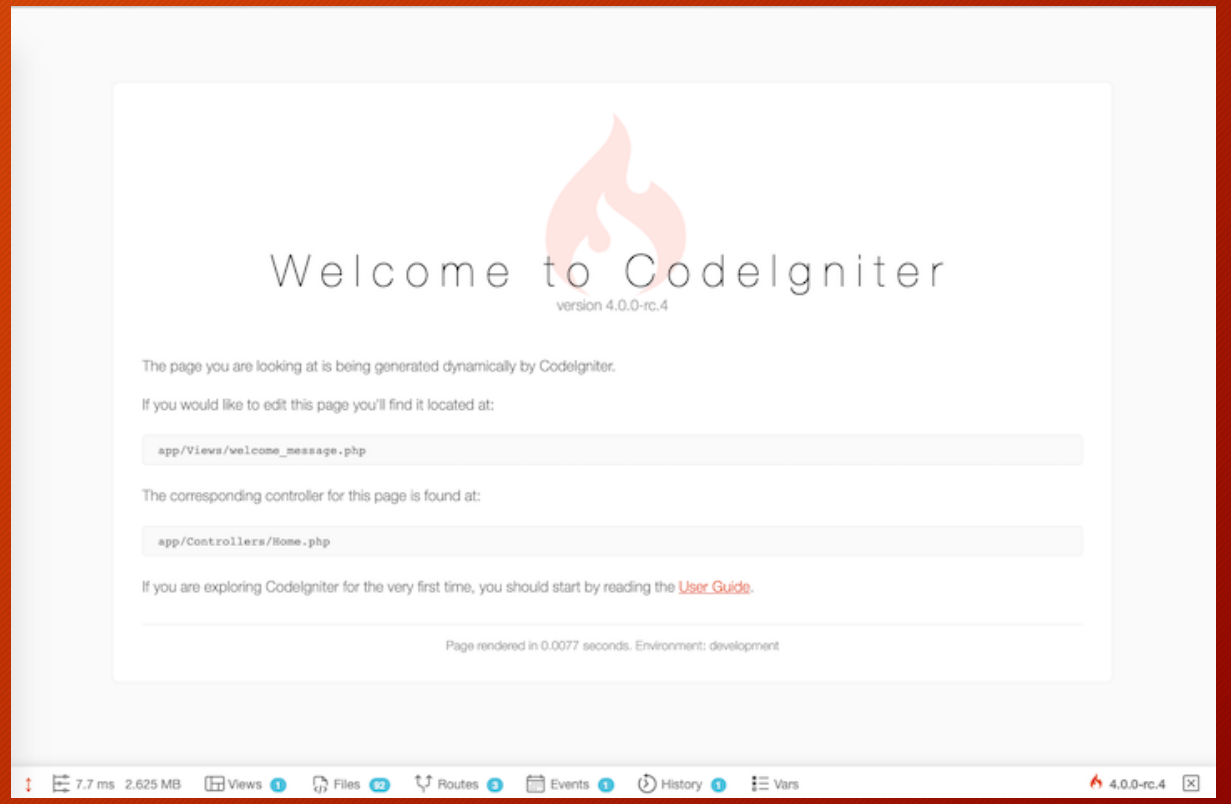
- Windows

  c:\> cd project

  c:\project> php spark serve

Codeigniter4 접속

- http://localhost:8080



USER API 개요

- 사용자 등록 : Create User - POST
- 사용자 조회 : Read User - GET
- 사용자 수정 : Update User - PUT
- 사용자 삭제 : Delete User - DELETE

USER API URL

- URL : http://localhost:8080/api/user
- 서버 : http://localhost
- 포트 : 8080
- PATH : api
- Resource : user

USER API 디자인

Method	URL	설명
POST	/api/user	user 생성
GET	/api/user	user list 조회
GET	/api/user/1	user 상세정보 조회
PUT	/api/user/1	user 수정
DELETE	/api/user/1	user 삭제

Model

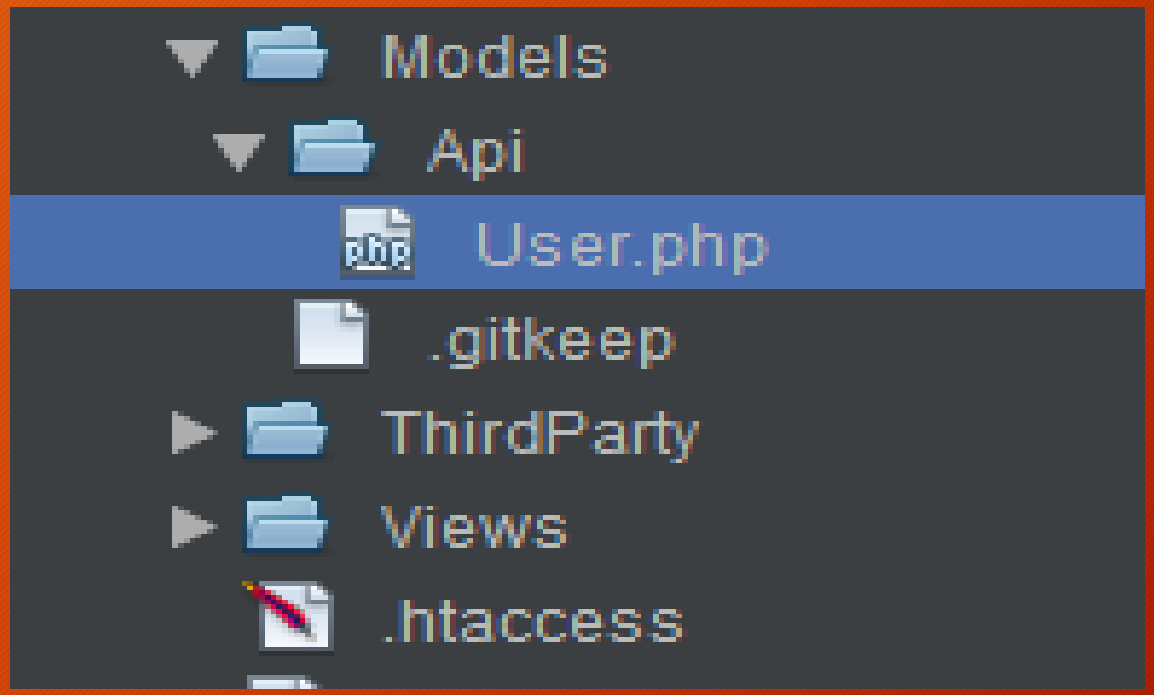
모델 만들기

CodeIgniter의 모델을 활용하려면 `CodeIgniter\Model`을 확장하는 새로운 모델 클래스를 만들면됩니다.

```php
<?php namespace App\Models;

use CodeIgniter\Model;

class UserModel extends Model
{

}
```

이렇게 작성된 클래스는 데이터베이스 연결, 쿼리 빌더 등 여러 가지 편리한 추가 메소드를 제공합니다.

User Model 만들기



User Model 만들기

```php
namespace App\Models\Api;

use CodeIgniter\Model;

/**
 * Description of User
 *
 * @author hoksi
 */
class User extends Model
```

User Model 만들기

```php
class User extends Model
{
    protected $table = 'users'; // 테이블 명
    protected $primaryKey = 'user_ix'; // primary key

    protected $useTimestamps = true;
    protected $createdField  = 'created_at'; // 생성일시
    protected $updatedField  = 'updated_at'; // 수정일시

    protected $allowedFields = ['user_id', 'user_pw', 'name']; // 수정 가능한 컬럼
}
```

컨트롤러

컨트롤러란 무엇입니까?

컨트롤러는 URI와 연결될 수 있는 방식으로 이름 붙여진 클래스 파일입니다.

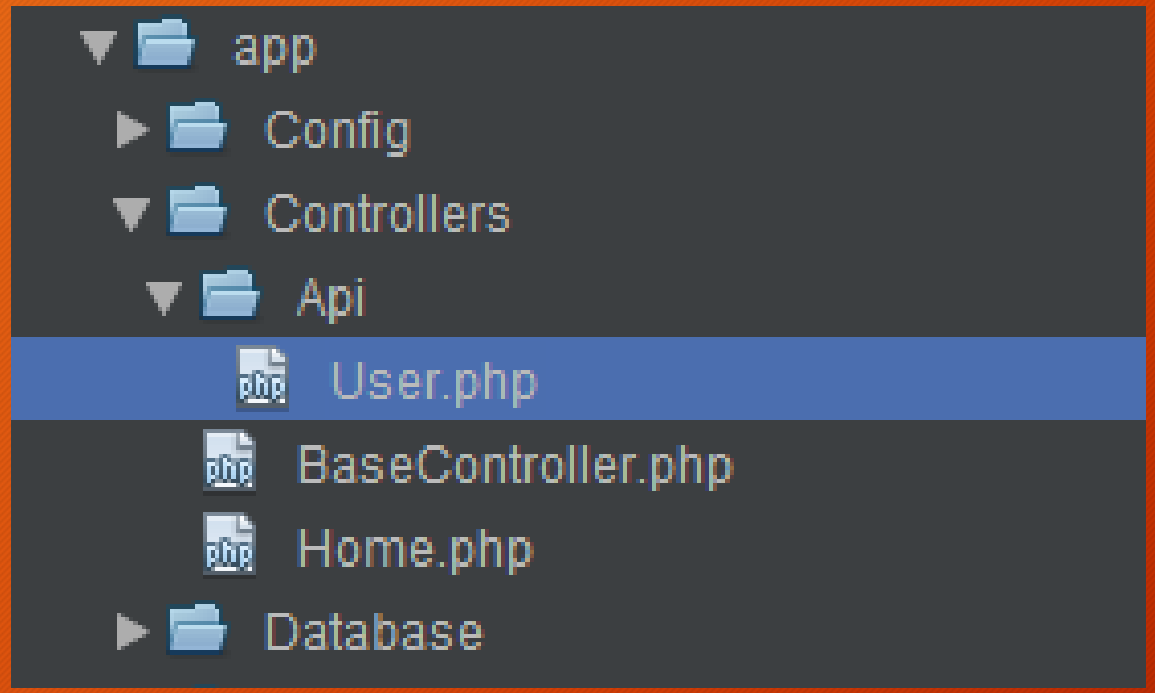
다음 URI를 살펴보세요.

```
example.com/index.php/helloworld/
```

위의 예제에서 CodeIgniter는 Helloworld.php 라는 컨트롤러를 찾아 로드하려고 시도합니다.

컨트롤러 이름이 URI의 첫 번째 세그먼트와 일치하면 로드됩니다.

API 컨트롤러 만들기



API 접근 URL

http://myhome.com/api

API 컨트롤러 만들기

```
namespace App₩Controllers₩Api;

use CodeIgniter₩API₩ResponseTrait;

class User extends ₩CodeIgniter₩Controller {
```

API Response Trait

API Response Trait

현대 PHP 개발의 대부분은 자바스크립트로 작성된 단일 페이지 어플리케이션(single page application)에 데이터를 제공하거나 독립형 제품으로 API를 구축하는 것입니다. CodeIgniter는 어떤 응답 유형에 대해 어떤 HTTP 상태 코드를 반환해야 하는지 기억할 필요없이 일반적인 응답 유형을 단순하게 만들기 위해 모든 컨트롤러와 함께 사용할 수 있는 API 응답 특성(trait)을 제공합니다.

사용샘플

다음 예는 컨트롤러내에서 일반적인 사용 패턴을 보여줍니다.

```php
<?php namespace App\Controllers;

use CodeIgniter\API\ResponseTrait;

class Users extends \CodeIgniter\Controller
{
```

API 컨트롤러 만들기

```php
class User extends \CodeIgniter\Controller {

    use ResponseTrait;

    protected $userModel;
    protected $data;

    public function __construct() {
        $this->userModel = new \App\Models\Api\User();
    }
```

API 컨트롤러 만들기 (POST)

```php
protected function post() {
    $ret = false;
    if (!empty($this->data)) {
        $ret = true;
        $this->userModel->insert($this->data);
    }

    return $this->respond($ret);
}
```

API 컨트롤러 만들기 (GET)

```php
protected function get($id = false) {
    if ($id) {
        $data = $this->userModel->find($id);
    } else {
        $data = $this->userModel->findAll();
    }

    return $this->respond($data);
}
```

API 컨트롤러 만들기 (PUT)

```php
protected function put($id = false) {
    $ret = false;
    if ($id && !empty($this->data)) {
        $ret = true;
        $this->userModel->update($id, $this->data);
    }

    return $this->respond($ret);
}
```

API 컨트롤러 만들기 (DELETE)

```php
protected function delete($id = false) {
    $ret = false;

    if ($id) {
        $ret = true;
        $this->userModel->delete($id);
    }

    return $this->respond($ret);
}
```

컨트롤러 매핑 재정의

리매핑 메소드 호출

위에서 언급 한 바와 같이, URI의 두 번째 세그먼트는 일반적으로 컨트롤러에서 호출되는 메소드를 결정합니다.
동작을 재정의 할 수 있습니다.

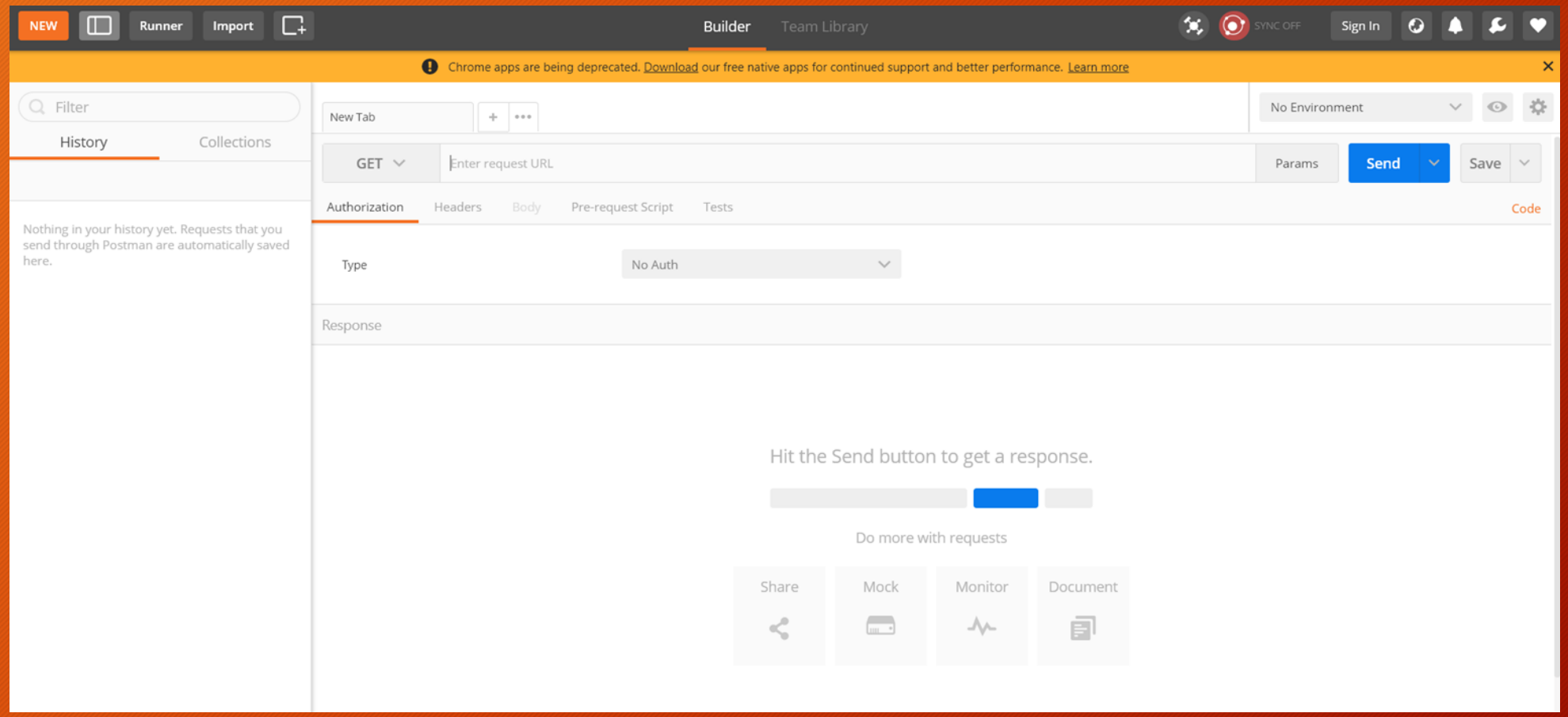
```
public function _remap()
{
        // Some code here...
}
```

API 컨트롤러 만들기 (_remap)

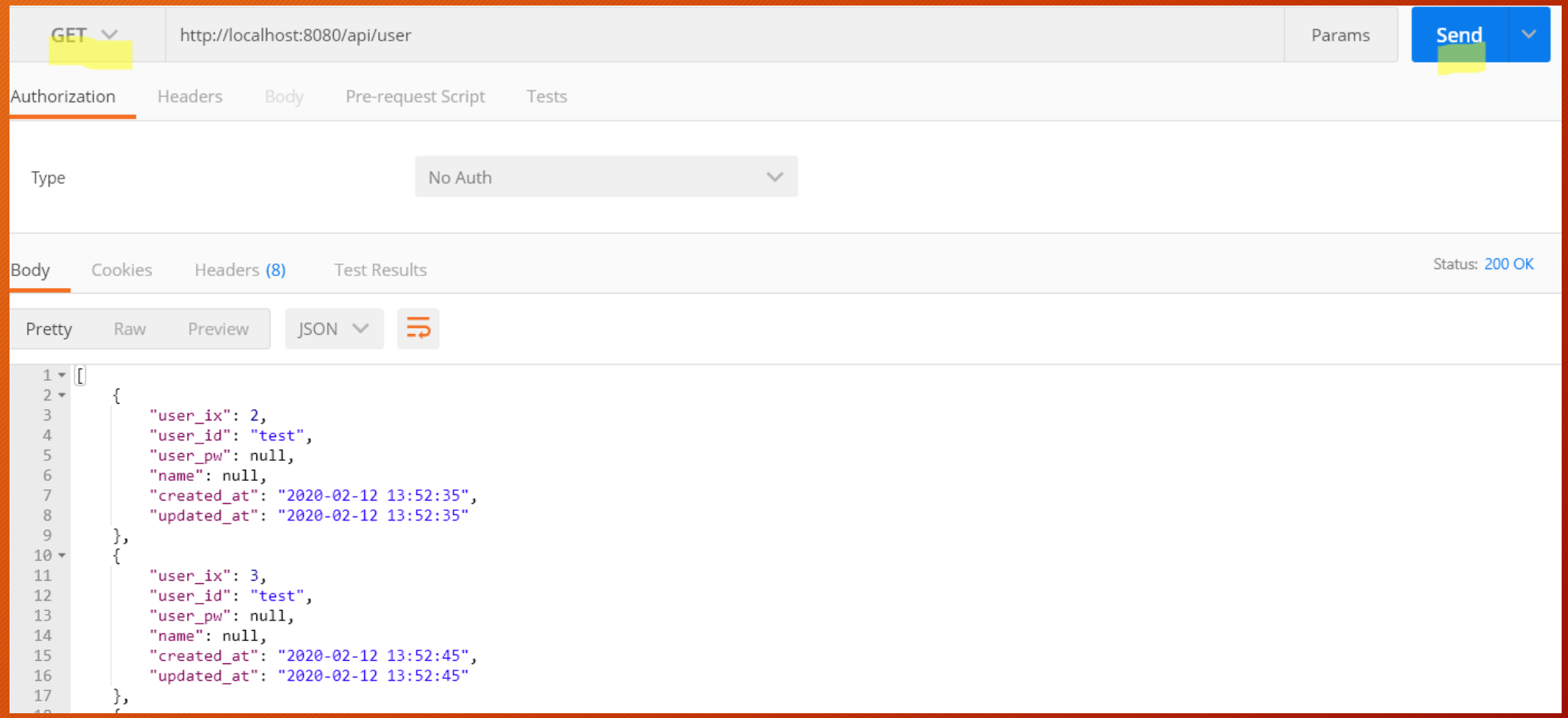
```php
public function _remap(...$params) {
    $method = $this->request->getMethod();
    $params = [($params[0] !== 'index' ? $params[0] : false)];
    $this->data = $this->request->getJSON();

    if (method_exists($this, $method)) {
        return call_user_func_array([$this, $method], $params);
    } else {
        throw \CodeIgniter\Exceptions\PageNotFoundException::forPageNotFound();
    }
}
```

RESTful API 테스트

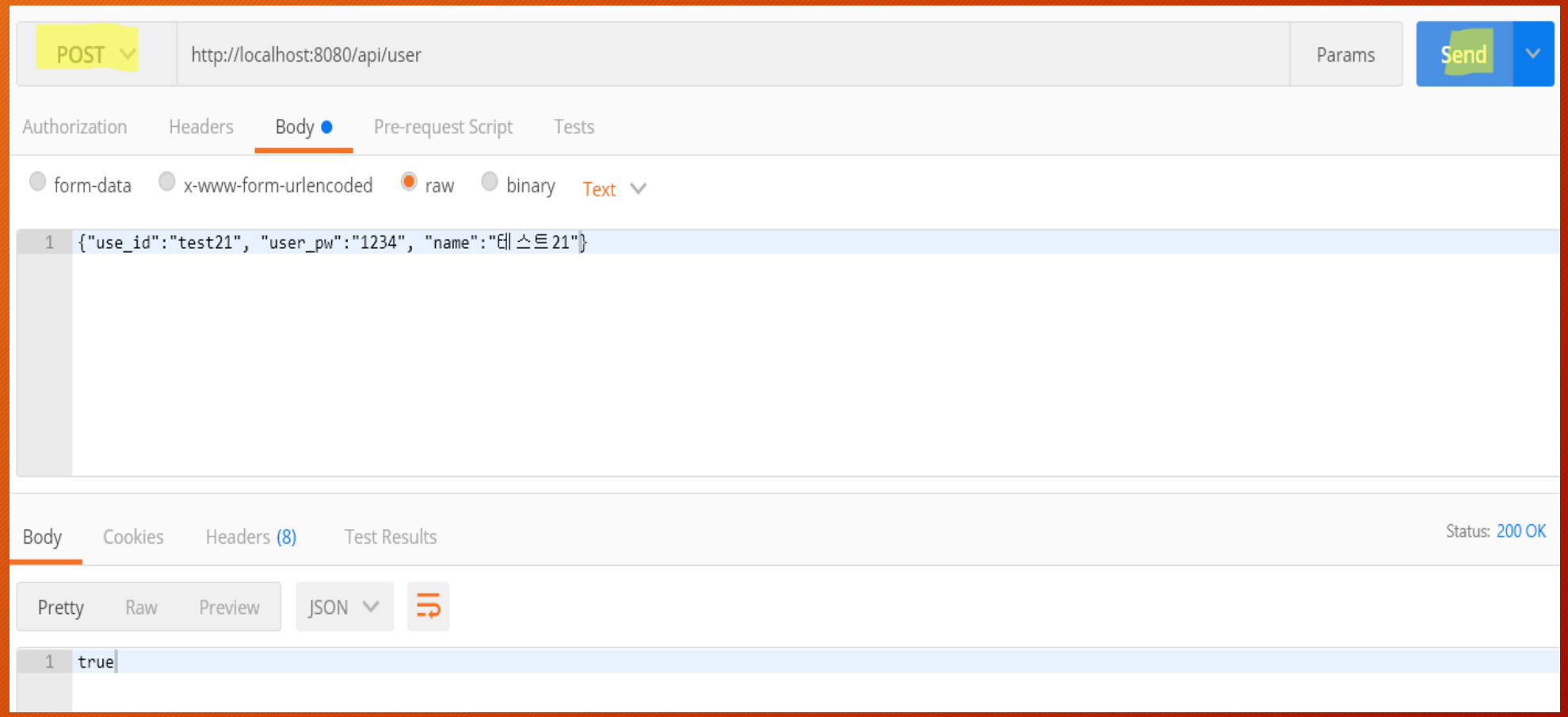
REST Client 실행



GET 테스트


GET
http://localhost:8080/api/user
Params
Send
Authorization
Headers
Body
Pre-request Script
Tests
Type
No Auth
Body
Cookies
Headers (8)
Test Results
Status: 200 OK
Pretty
Raw
Preview
JSON
[
{
"user_ix": 2,
"user_id": "test",
"user_pw": null,
"name": null,
"created_at": "2020-02-12 13:52:35",
"updated_at": "2020-02-12 13:52:35"
},
{
"user_ix": 3,
"user_id": "test",
"user_pw": null,
"name": null,
"created_at": "2020-02-12 13:52:45",
"updated_at": "2020-02-12 13:52:45"
},

POST 테스트

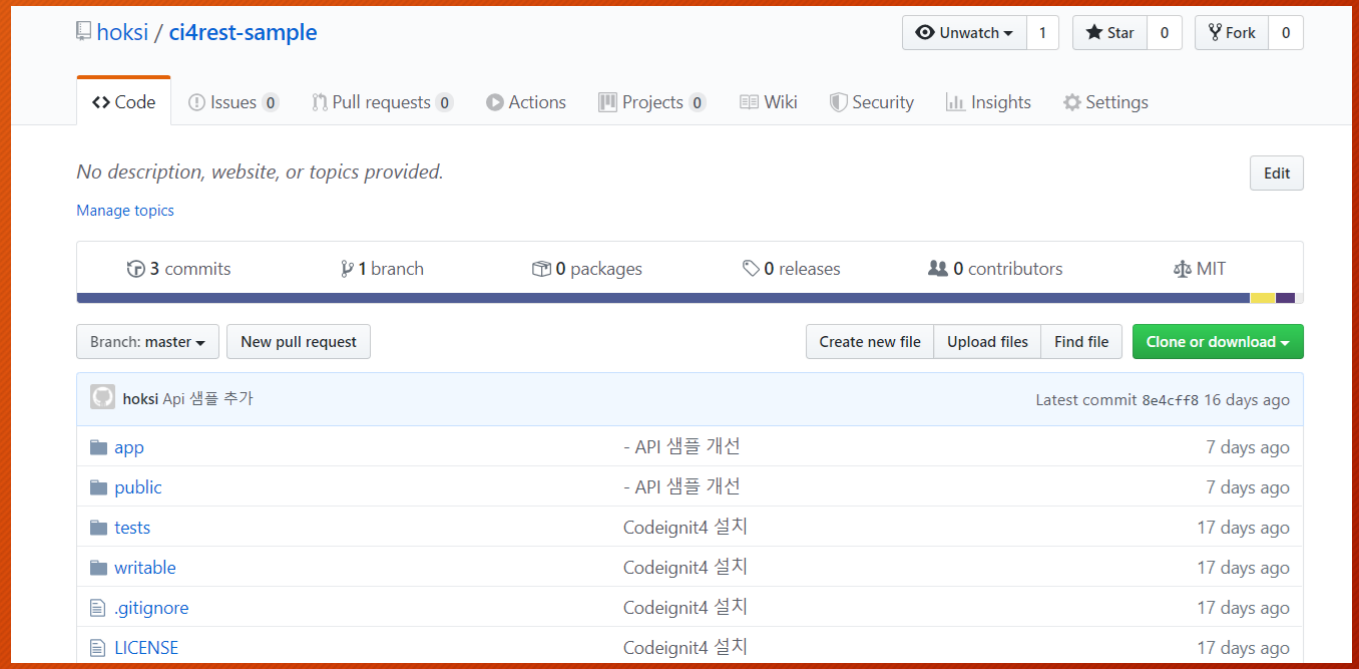

POST
http://localhost:8080/api/user
Params
Send
Authorization
Headers
Body
Pre-request Script
Tests
form-data
x-www-form-urlencoded
raw
binary
Text
1 {"use_id":"test21", "user_pw":"1234", "name":"테스트21"}
Body
Cookies
Headers (8)
Test Results
Status: 200 OK
Pretty
Raw
Preview
JSON
1 true

POST 테스트


```
    },
    {
        "user_ix": 6,
        "user_id": "test",
        "user_pw": null,
        "name": null,
        "created_at": "2020-02-12 14:24:09",
        "updated_at": "2020-02-12 14:24:09"
    },
    {
        "user_ix": 7,
        "user_id": null,
        "user_pw": "1234",
        "name": "테스트21",
        "created_at": "2020-02-19 06:44:57",
        "updated_at": "2020-02-19 06:44:57"
    }
]
```

API 샘플 코드 다운로드

https://github.com/hoksi/ci4rest-sample



Q & A

Email : hoksi3k@gmail.com

코드이그나이터 한국 사용자 포럼

https://cikorea.net

감사합니다

시 연