furkanmeclis/php-router

1 Başlangıç

Php İçin Dil Destekli Yönlendirme Sınıfı.

Özellikler

- GET,POST,PUT ve DELETE istek metotları destekleniyor.
- Controller dosyaları destekleniyor.
- Middleware kontrolü yapılabiliyor.
- Özelleştirilmiş parametreler destekleniyor.
- Yeni doğrulama deseni eklenebiliyor.
- Namespace desteği mevcut.
- Gruplama özelliği mevcut.
- Dil desteği mevcut.
- Özelleştirilmiş hata sayfaları.

Yükleme

- Composer İle Yükleme

```
composer require furkanmeclis/router
```

- Manuel Olarak yükleme

`src/Router.php` Dosyasını indirerek projenize dahil edebilirsiniz.

2 Metotlar

Bu sınıf 4 istek metodunu destekliyor.

- GET
- POST
- PUT
- DELETE

Örnek kullanım

```
<?php
    require './vendor/autoload.php';
    $router = new furkanmeclis\Router([
        "namespaces" => [
            "controller" => 'App\Controller\\',
            "middleware" => 'App\Middleware\\'
        ],
        "paths" => [
        "controller" => 'App/Controller/',
        "middleware" => 'App/Middleware/'
        ],
        "error" => [
            "controller" => "Home",
            "method" => "error"
        ],
        "language" => [
            "default_language" => "tr",
            "router_file_url" => "/router.json"
        ]
    ]);

    $router->get('/api/users','ApiController@users');

    $router->post('/api/adduser','ApiController@adduser');

    $router->put('/api/updateuser','ApiController@updateuser');

    $router->delete('/api/deleteuser','ApiController@deleteuser');

?>
```

3 Parametre Desenleri

Mevcut Desenler

- `:all` => Tüm karakterler
- `:any` => '/' karakteri olmayan tüm karakterler
- `:id` => Doğal sayılar (0, 1, 17, 167, 1881, 20956...)
- `:number` => Gerçek sayılar (1, -1.2, 5.5 ...)
- `:float` => Gerçek sayılar
- `:bool` => Boole değerleri. (doğru, yanlış, 1, 0)
- `:string` => Alphanumeric characters
- `:slug` => SEO için URL biçimi karakterleri. (Alfanümerik karakterler, _ ve -)
- `:uuid` => UUID
- `:date` => Y-m-d biçimi tarih dizesi

Yeni Desen Ekleme

```
$router = new furkanmeclis\Router();

$router->get('/api/:desen')->where('desen','[\w\-_]+');
```

- where(**$key**,**$pattern**)
  - $key => Desenin ismi
  - $pattern => Desen başında ve sonunda parantez olmadan

4 Controllers

Route tanımlamalarımızı yaparken controller dosyalarına ihtiyacımız olacaktır. Sınıfı başlatırken ;

- paths
- namespaces adında iki tane dizi vardır bunlar içine şunları alır;
- controller
- middleware Bunlar ise şu şekilde kullanılır

```
$router = new furkanmeclis\Router([
        "namespaces" => [
            "controller" => 'App\Controller\\',
            "middleware" => 'App\Middleware\\'
        ],
        "paths" => [
        "controller" => 'App/Controller/',
        "middleware" => 'App/Middleware/'
        ]
    ]);
```

Sınıfımız controller dosyasını çağırırken `/App/Controller/` altında arar. Route tanımlaması yaparken dosya adı değil sınıf ismi yazılır.

- App/Controller/Api.php

```
<?php
namespace App\Controller;
class ApiController{
    public function home(){
        echo "merhaba";
    }
}
?>
```

- index.php

```
$router->get('/home','ApiController@home'); // merhaba
```

5 Middlewares

Middleware Nedir ?

route tanımlaması yaparken bazı tanımlamalar herkesin erişmemesi gereken yönlendirmeler olabilir böyle durumlarda **Middleware** karşımıza çıkıyor.

Nasıl Kullanılır ?

Middleware sınıfının çağrılması için sınıf çağrılırken gerekli tanımlamalar yapılmalı.

```
$router = furkanmeclis\Router([
            "namespaces" => [
                "controller" => 'App\Controller\\',
                "middleware" => 'App\Middleware\\'
             ],
             "paths" => [
                "controller" => 'App/Controller/',
                "middleware" => 'App/Middleware/'
             ],
         ]);
```

- index.php

```
$router->get('/home',function(){
   echo "/home";
},'TestAuth');
```

- App/Middleware/TestMiddleware.php

```
<?php
namespace App\Middleware;
class TestAuth{
    public function handle(){
        if(isset($_SESSION['login']) || $_SESSION['login'] == true){
            return true;
        }else{
            return false;
        }
    }
}
?>
```

Eğerki handle fonksiyonundan `true` dönerse sayfa gösterilir ancak `false` dönerse sayfa gösterilemez.

Global Middleware Nasıl Tanımlanır

```
$router->setGlobalMiddleware('TestAuth');
```

6 Gruplama

Route tanımlaması yaparken mesela api için bir tanımlama yapmak istiyorsunuz yollarımız bunlar olsun;

- `/api/getuser` - (**get**)
- `/api/adduser` - (**post**)
- `/api/deleteuser` - (**delete**)
- `/api/updateuser` - (**put**) Tanımlamalarda şu şekilde olsun;

```
$router->get('/api/getuser','ApiController@getuser','TestAuth');
$router->post('/api/adduser','ApiController@adduser','TestAuth');
$router->delete('/api/deleteuser','ApiController@deleteuser','TestAuth');
$router->put('/api/updateuser','ApiController@updateuser','TestAuth');
```

farkettiyseniz hepsinde `/api` mevcut hem işimizin kolaylaşması için hemde daha toplu bir kod yapısı için karşımıza **`group()`** fonksiyonu çıkıyor.

Group() Nasıl Kullanılır ?

Group() fonksiyonu üç tane parametre alır bunlar;

- $prefix(zorunlu)
- $callback(zorunlu)
- $middleware(varsayılan olarak `false`)

Yukarıdaki örneği birde `group()` ile yapalım;

```
$router->group('/api',function($r){
   $r->get('/getuser','ApiController@getuser');
   $r->post('/adduser','ApiController@adduser');
   $r->delete('/deleteuser','ApiController@deleteuser');
   $r->put('/updateuser','ApiController@updateuser');
},'TestAuth');
```

Kullanımı bu kadar kolay.

7 Kişiselleştirilmiş Hata Sayfaları

Sınıfı kullanırken bazı hatalarla karşılaşabilirsiniz bu hatalar şunlardır;

- **Page Not Found**(Sayfa Bulunamadı[404])
- **Controller Not Found**(Controller Bulunamadı)
- **Method Not Found**(Method Bulunamadı)
- **No Route Defined**(Route Tanımlaması Yapılmamış) Bu hataları kişiselleştirilmiş bir sayfada göstermek istiyorsanız karşınıza `error()` fonksiyonu çıkıyor. `error(` **$classname** `)`
  - $classname Hata hangi sınıfta gösterilecek örn. `Home@error`

Örnek Kullanım

- index.php

```
$router->error('Home@error');
```

- App/Controller/Home.php

```php
<?php
namespace App\Controller;
class Home{
    public function error($text){
          echo "<b style='color:red'>$text</b>";
    }
}
?>
```

8 Dil Sistemi

Eğer geliştirdiğiniz proje birden fazla dile sahip ise ve route tanımı yapmak istiyorsanız bu fonksiyonlar tam size göre;

- `initLanguage($routearray)`
- `setLanguage($lang)`
- `getActiveLanguage()`
- `setDefaultLanguage()`
- `getLink($link)`
- `language($middleware = false)`

1. **initLanguage($routearray)**
    - Dil sistemini başlatır içine bir array alır ve arraydeki verileri .json'a dönüştürüp statik olarak dizine ekler
    - Tanımlamaları yaparken aynı sayfaların keyleri aynı olmalı
    - Örnek Kullanım

```php
$router->initLanguage([
    "tr" => [
         "home" => ["anasayfa","Homecontroller@home"],
         "contact" =>["iletisim","Homecontroller@contact"]
     ],
     "en" => [
          "home" => ["home","Homecontroller@home"],
          "contact" =>["contact","Homecontroller@contact"]
     ]
 ]);
```

2. **setLanguage($lang)**
    - $lang initLanguage() fonksiyonundaki dil kısaltmaları ile dili değiştirir
3. **getActiveLanguage()**

- Aktif dili geri dönderir örn. `tr`

4. **setDefaultLanguage()**

   - Varsayılan dili ayarlar.Varsayılan dil ise sınıf başlatılırken `"language" => ["default_language" => "tr"]` şeklinde ayarlanır.

5. **getLink($link)**

   - dil routes içerisinden aktif dile ait dizeden $link keyine ait tanımlamayı linke dönderir
   - Örneğin getLink('contact') eğer dil türkçe ise `/tr/iletisim` , ingilizce ise `/en/contact` geri dönderir

6. **language($middleware = false)**

   - Dil tanımlamalarını yönlendirme sınıfına aktarır
   - $middleware eğer middleware kontrolüne ihtiyacınız var ise buraya middleware tanımlaması yapabilirsiniz