طریق داشتن dependency ها:

مثلا دو کلاس A و B داریم وقتی میخواهیم متد های داخل کلاس B را در A استفاده میکنیم.

service provider و Service Container

service providers → ارائه دهندهٔ سرویس ها می باشد. و راه اندازه می کند.

وظیفهٔ اصلی service provider این است که

کدهای مختلف را گروه بندی می کند و آنها را Run می کند.

service Container → جایی که تمام service ها ثبت نام شده است.

سرویس ها را به container اضافه می کند.

مثلا مقدار بیشترین یک ماشین را بخواهد و بگوید به ازای هر لیتر ... کیلومتر می رود.

دستور یک فایل درست می کنیم

app → service → php. اسم فایل

api ← با استفاده از یک برنامه خارجی خارج از برنامه ای که درآن کد می‌نویسیم استفاده کنیم.

با پایتون می‌توانیم سرویس دهنده و سرویس گیرنده باشیم.

مهم:

○ تایپ دیتا ← باید برای هردوتا سیستم قابل شناسایی باشد و معمولاً از نوع json است. (xml)

○ احراز هویت ← authication

rest api ← یک معماری است که میگه این api هایی که تولید می‌شن باید در بستر http باشد. و باید از متدهایی که مورد قبول ماست استفاده بشن.

منظور httpware هاست یعنی get، post، put، delete.

task	method	path
ایجاد یک عمل جدید مثلا برای مشتری → create	post	/customers
برای حذف کردن → delete	delete	/customers/{id}
براساس id چیزی را پیدا می‌کنه → find()	get	/customers/{id}
همه‌ی مقادیر را میاورد → all()	get	/customers
بروزرسانی	put	/customers/{id}

CS CamScanner

سه‌شنبه
19 September
۳ ربیع‌الاول
۲۸

ساخت api : کنترلر ساخته می‌شوند

php artisan make: controller Api\postsController --api

app → Http → Api → ApiPosts controller.php

دقیقا شبیه resource controller هایی است که داشتیم است -
با فرق این است که create و edite ندارد .

در واقع ما در api ها صفحه نداریم و عملیات را انجام می‌دهیم این دو متد وجود ندارد .

روت دهی برای استفاده از api ها :

route → api.php   مسیرهای api داخل api.php است .

داخل api.php :

```
Route::namespace('Api')->group(function(){

    Route::apiResource('posts','Api Posts controller');
```

مربوط به کدام کنترلر است

داخل کنترلر api ←

```
public function index(){
  return post::all();}

public function store(Request $request){
  return post::create(['title'=>'api','user_id'=>1]);}

public function show($id){
  return post::findOrFail(id);}

public function update(Request $request,$id)
{ $post = post::findOrFail($id);
  $post->update($request->only('title','api2'));

public function destroy($id){

  Post::findOrFail($id)->delete(); }
```

۱۴۰۲
2023
۱۴۴۵

ارسال فرم داخل postman : در update استفاده می‌شود

داخل تب params ← شامل جدولی است که key و value دارد

روش ۱: در آن مقادیر را می‌نویسیم

key	value
key مقدار	value مقدار

نوع URL از نوع put

روش ۲: داخل تب body ← ارسال فرم تب form-data :

نکته در روش ۲ حتما باید در تب headers

key	value
Accept	application/json

در تب form-data :

key	value	details
title	api2	
_method	put →	مشخص کرده put است

همچنین در URL تست postman نوع باید از نوع post باشد.

پنجشنبه
21 September
۵ ربیع‌الاول
۳۰

```
public function index(){
return post::paginate(5);}
```

get http://localhost:8000/api/ : postman در
send

params ←
key page
value 5 → صفحه پنجم

```
public function index(Request $request){
$sort = $request->input('sort', 'id');
$sortDirection = Str::startsWith($sort, '-') ? 'desc' : 'asc';
use Illuminate\Support\Str;
$sort = ltrim($sort, '-');
return post::orderBy($sort, $sortDirection)->paginate(5);
```

get http://localhost:8000/post
params
key
value
sort

Apl Resources : برای تبدیل به متن استفاده می‌شود.

۱) ابتدا برای مثال یک رابطه ۱ به n بین مدل‌های post و comment نوشته

۲) حالا برای هر مدل یک Resource (منبع) می‌سازیم.

php artisan make:resaurce اسم مدل

→ app → Http → ساخته می‌شود.

```
public function toArray ($request) {

Return [
        'id' => $this->id,
        'title' => $this->title, ];
```

در کنترلر api :
```
use app\http\Resources\post as postResource;
public function index(Request $request){

return new postResource (post::find(78));}
```

Resources آرایه را به صورت متن json برگشت می‌دهد.

ارسال فرم داخل postman : در update استفاده می‌شود

داخل گزینه params ← شامل جدولی است که key و value دارد

روش ۱: درآن مقادیر را می‌نویسیم

key	value
key مقدار	value مقدار

نوع URL از نوع put

روش ۲: داخل گزینه body ← ارسال فرم گزینه form-data :

نکته در روش ۲ حتما باید در گزینه headers

key	value
Accept	application/json

در گزینه form-data :

برای delete

key	value
title	api2
-method	put → مشخص کردیم put است

همچنین در URL سمت postman نوع باید از نوع post باشد.

نکته وقتی داریم از form مقداری را به کنترلر ارسال می‌کنیم حتما نوع URL را از نوع post می‌گذاریم و درست form-data :

key	value
-method	put ← باید باشد

درغیر این صورت مقداری که از postman به کنترلر ارسال می‌شد دارای خطا است.

عملیات صفحه‌بندی یعنی اگر ۲۰ صفحه داشته باشیم
می‌خواهیم آنها را صفحه‌بندی کنیم و مثلا ۵ تا ۵ تا نمایش بدهیم داخل
کنترلر api

```
public function index(){
  return post::paginate(5);}
```

get http://localhost:8000/api/ : مثلا در postman
sendر

params ← می‌بینیم

key	value
page	5 → مثلا ۵ صفحه بعدی یا صفحه ۵ام

نحوه عملیات Sorting یعنی در api چگونه انجام دهیم.

در کنترلر api:

```
public function index(Request $request){
  $sort = $request->input('sort', 'id');
  $sortDirection = Str::startsWith($sort, '-') ? 'desc' : 'asc';
use Illuminate\support\Str;
  $sort = ltrim($sort, '-');
  return post::orderBy($sort, $sortdirection)->paginate(5);
```

Str::startsWith ← باید بالا use شود

مثلا اگر $sort با منفی شروع شد یعنی نمایش در غیر این صورت افزایشی

ltrim ← منفی را حذف می‌کند از sort

در postman: get http://localhost:8000/post
params
key → sort
value → cde

API Resources : برای تبدیلات به جیسون استفاده می‌شود.

۱. ابتدا برای مثال یک رابطه ای بین مدل های post و comment نوشته

۲. حالا برای هر مدل یک Resource (منبع) می‌سازیم.

```
php artisan make:resource اسم مدل
```

برای دو مدل post و comment ← app → http → ساخته می‌شود.

```
public function toArray ($request) {
  Return[
    'id' => $this->id,
    'title' => $this->title, ];
```

بیرون آوردن فقط برای از مقدار استفاده می‌شود.

۳. در کنترلر api :

```
use app\http\Resources\post as postResource;
public function index(Request $request){
return new postResource(post::find(78));}
```

برای یک مقدار

Resources این داده‌ها را دریافت می‌کند و به جیسون تبدیل می‌دهد.

مشاهده در postman.

نکته : دستور زیر مقادیری که داخلش قرار دارند را در خروجی مخفی می‌کند

```
protected $hidden[
  'id' ];
```

دستور زیر مقادیری که داخلش هستند را نشان می‌دهد در خروجی

```
protected $visible[
  'id' ];
```


استفاده از collections : برای دریافت همه مقادیر استفاده می‌شود

دقیقا برای Resources های که ساختیم شامل post و comment است

باید collection ها را بسازیم.

```
php artisan make:resource اسم مدل --collection
```

برای نمایش دادن کل تمام post ها :

داخل post collection :

```
public function toArray($request){
return[
    'data' => $this->collection, ];
```

داخل کنترلر Api :

```
public function index(Request $request){
return new postCollection(post::all());}
```

با پوشش انجام شود. برای همه یا همه ستون‌ها

داخل postman :

```
get http://localhost:8000/api/posts
```

عملیات login به همراه api:

برای اینکه متوجه بشویم کاربر login کرده باید در جدول users یک فیلد به اسم api token اضافه کنیم. از نوع char

```
$table->char('api_token', 60)->nullable();
```

هر موقع کاربر login کرد فیلد api_token یک مقدار تصادفی میگیرد و وقتی logout میشود مقدارش null میشود.

۲) در route های api:

```
Route::post('login', function(Request $request){
  if(auth()->attempt(['email'=>$request->input('email'), 'password'=>$request->
  input('password')])){
    $user = auth()->user();   → یعنی یوزر ساخته شده
    $user->api_token = Str::random(60);
    $user->save();
    return $user; }
  return response()->json([
    'error'=> 'غیر مربوطه' ], 401);
```

(اگر لاگین شد: بلوک بالا — اگر لاگین نشد ⇐ return response()...)

۳) در postman:

post http://localhost:8000/api/login — برای اینکه اطلاعات درست ارسال بشه

در قسمت headers

	key	value
✓	Accept	application/json

در قسمت login ← form-data

key	value
email	مقدار email
password	مقدار password

عملیات logout در api:

از gard استفاده میشود حال api_token

1 در روت دهی مینویسیم:

```
Route::middleware('auth:api')->post('logout', function (Request $request)
{

if (auth()->user()){

    $user = auth()->user();
    $user->api_token = null;
    $user->save();

    return response()->json([

      'message'=>'logout' ]); }

return response()->json([

    'error' =>'error' ]), 401); }
```

2 در postman

post http://localhost:8000/api/login
" " /logout

در قسمت authorization

type
bearer token token-> مقدار سطح api_token را قرار میدیم

در قسمت body

key	value
email	مقدار
password	"