



Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 6 (enam)
Pertemuan ke- : 11 (sebelas)
Nama : Karina Ika Indasa

JOBSHEET 11

RESTFUL API 2

Sebelumnya kita sudah membahas mengenai *RESTFUL API dan JSON Web Token (JWT)* pada Laravel. Dimana kita telah membuat RESTful API Register, RESTful API Login, RESTful API Logout, dan implementasi CRUD dalam RESTful API pada halaman web. Pada pertemuan kali ini, kita akan mempelajari penerapan RESTFUL API lanjutan di dalam project Laravel.

Sebelum kita masuk materi, kita buat dulu project baru yang akan kita gunakan untuk membangun aplikasi sederhana dengan topik *Point of Sales (PoS)*, sesuai dengan **Studi Kasus PWL.pdf**.
Jadi kita bikin project Laravel 10 dengan nama **PWL_POS**.

Project PWL_POS akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

A. ELOQUENT ACCESSOR

Laravel memiliki fitur yang bernama mutator, accessor dan casting, fitur-fitur ini digunakan untuk melakukan manipulasi data di dalam attribute database dengan sangat mudah. Contohnya pada saat insert data dengan enkripsi ke dalam database dan melakukan deskripsi saat menampilkan dari database secara otomatis.

Accessor dapat mengubah nilai saat attribute atau field eloquent diakses. Untuk mendefinisikan Accessor dapat membuat method di dalam model untuk menentukan attribute yang akan diakses. Nama method yang dibuat harus sama dengan nama attribute yang akan di format: contoh :

Attribute/field pada tabel `first_name` maka methodnya `firstName()`



protected function firstName(): Attribute

```
{  
    //...  
}
```

Jika membuat attribute/field image yang ada di table m_user kita akan memberikan nilai full path dari direktori dimana file gambar tersebut disimpan. contohnya pada UserModel ditambahkan

```
protected function image(): Attribute  
{  
    return Attribute::make(  
        get: fn ($image) => url('/storage/posts/' . $image),  
    );  
}
```

Dengan begitu dapat melakukan import Eloquent Attribute dengan

```
use Illuminate\Database\Eloquent\Casts\Attribute;
```

Lalu method baru dengan nama image() melakukan return path nama file image itu berada

```
get: fn ($image) => url('/storage/posts/' . $image),
```

Hasil akhir memanggil attribute image

```
domain.com/storage/posts/nama_file_image.png
```

Praktikum 1 – Implementasi Eloquent Accessor

1. Sebelum memulai pastikan REST API, terlebih dahulu pastikan sudah ter instal aplikasi Postman.
2. Kita akan memodifikasi Table m_user dengan menambahkan column : image, buka terminal lalu ketikkan

php artisan make:migration add_image_to_m_user_table

```
C:\laragon\www\PWL_2025\Week11\JS11>php artisan make:migration add_image_to_m_user_table
```

```
INFO Migration [C:\laragon\www\PWL_2025\Week11\JS11\database\Migrations\2025_05_04_121024_add_image_to_m_user_table.  
php] created successfully.
```

3. Buka file migrasi tersebut lalu modifikasi seperti ini lalu simpan:



```
JS11 > database > migrations > 2025_05_04_121024_add_image_to_m_user_table.php > ...
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration {
8      /**
9       * Run the migrations.
10      */
11     public function up(): void
12     {
13         Schema::table('m_user', callback: function (Blueprint $table): void {
14             $table->string(column: 'image');
15         });
16     }
17
18     /**
19      * Reverse the migrations.
20      */
21     public function down(): void
22     {
23         Schema::table('m_user', callback: function (Blueprint $table): void {
24             $table->dropColumn(columns: 'image');
25         });
26     }
27 };
```

4. Lakukan jalankan update migrasi dengan cara: php artisan migrate

```
C:\laragon\www\PWL_2025\Week11\JS11>php artisan migrate
INFO Running migrations.
2025_05_04_121024_add_image_to_m_user_table ..... 131ms DONE
```

5. Lalu lakukan modifikasi models pada App/Models/UserModel.php

```
JS11 > app > Models > UserModel.php > PHP > UserModel
1  <?php
2  namespace App\Models;
3
4  use Illuminate\Database\Eloquent\Model;
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Relations\BelongsTo;
7  use Illuminate\Foundation\Auth\User as Authenticatable;
8  use Tymon\JWTAuth\Contracts\JWTSubject;
9  use Illuminate\Database\Eloquent\Casts\Attribute;
10
11  43 references | 0 implementations
12  class UserModel extends Authenticatable implements JWTSubject
13  {
14      use HasFactory;
15
16      0 references | 0 overrides
17      public function getJWTIdentifier(): mixed
18      {
19          return $this->getKey();
20      }
21
22      0 references | 0 overrides
23      public function getJWTCustomClaims(): array
24      {
25          return [];
26      }
27
28      0 references
29      protected $table = 'm_user';
30
31      0 references
32      protected $primaryKey = 'user_id';
33
34      0 references
35      protected $fillable = ['username', 'password', 'nama', 'level_id', 'profile_picture', 'created_at', 'updated_at', 'image']; //tambahan
36
37      0 references
38      protected $hidden = ['password'];
39
40      0 references
41      protected $casts = ['password' => 'hashed'];
42  }
```



```
33 /**  
34  * Relasi ke tabel level  
35  */  
36 @references | 0 overrides  
37 public function level(): BelongsTo  
38 {  
39     return $this->belongsTo(related: LevelModel::class, foreignKey: 'level_id', ownerKey: 'level_id');  
40 }  
41 /**  
42  * Mendapatkan nama role  
43  */  
44 @references | 0 overrides  
45 public function getRoleName(): string  
46 {  
47     return $this->level->level_nama;  
48 }  
49 /**  
50  * Cek apakah user memiliki role tertentu  
51  */  
52 @references | 0 overrides  
53 public function hasRole($role): bool  
54 {  
55     return $this->level->level_kode == $role;  
56 }  
57 /**  
58  * Mendapatkan nama role  
59  */  
60 @references | 0 overrides  
61 public function getRole(): mixed  
62 {  
63     return $this->level->level_kode;  
64 }  
65 @references | 0 overrides  
66 public function getProfilePictureUrlAttribute(): string  
67 {  
68     return $this->profile_picture  
69     ? asset(path: 'storage/profile_pictures/' . $this->profile_picture)  
70     : asset(path: 'images/default-profile.png');  
71 }  
72 @references | 0 overrides  
73 protected function image(): Attribute  
74 {  
75     return Attribute::make(  
76         get: fn($image): string|UrlGenerator => url(path: '/storage/posts/' . $image),  
77     );  
78 }
```

6. Lakukan modifikasi pada Controllers/Api/RegisterController

```
1511 > app > Http > Controllers > Api > RegisterController.php > PHP > RegisterController > __invoke()  
1 <?php  
2  
3 namespace App\Http\Controllers\Api;  
4  
5 use App\Models\UserModel;  
6 use App\Http\Controllers\Controller;  
7 use Illuminate\Http\Request;  
8 use Illuminate\Support\Facades\Validator;  
9  
10 @references | 0 implementations  
11 class RegisterController extends Controller  
12 {  
13     @references | 0 overrides  
14     public function __invoke(Request $request): JsonResponse|mixed  
15     {  
16         // Set validation  
17         $validator = Validator::make(data: $request->all(), rules: [  
18             'username' => 'required',  
19             'nama' => 'required',  
20             'password' => 'required|min:5|confirmed',  
21             'level_id' => 'required',  
22             'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048'  
23         ]);  
24  
25         // If validation fails  
26         if ($validator->fails()) {  
27             return response()->json(data: $validator->errors(), status: 422);  
28         }  
29  
30         // Create user  
31         $user = UserModel::create(attributes: [  
32             'username' => $request->username,  
33             'nama' => $request->nama,  
34             'password' => bcrypt(value: $request->password),  
35             'level_id' => $request->level_id,  
36             'image' => $request->image  
37         ]);  
38  
39         // Return response JSON if user is created  
40         if ($user) {  
41             return response()->json(data: [  
42                 'success' => true,  
43                 'user' => $user,  
44             ], status: 201);  
45         }  
46  
47         // Return JSON if process insert failed  
48         return response()->json(data: [  
49             'success' => false,  
50             ], status: 409);  
51     }  
52 }
```



7. Anda dapat menambahkan detail untuk spesifikasi image pada validator

```
20 | 'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048'
```

8. Ubah atau tambahkan register1 pada routes/api.php

```
24 | Route::post(uri: '/register1', action: App\Http\Controllers\Api\RegisterController::class)->name(name: 'register1');
```

9. Simpan dan akses pada aplikasi Postman, atur pada Body isi manual Key dan Valuenya pada Key image tambahkan value File dan upload gambar

<http://127.0.0.1:8000/api/register1> dengan method POST dan klik send

Postman interface showing a successful POST request to <http://127.0.0.1:8000/api/register1>. The request body is form-data with fields: username (penggunaempat), nama (pengguna4), password (12345), password_confirmation (12345), level_id (2), and image (icut.jpg). The response is JSON: {"success": true, "user": {"username": "penggunaempat", "nama": "pengguna4", "level_id": 2, "image": "http://127.0.0.1:8000/storage/posts/C:\\Users\\ASUS\\AppData\\Local\\Temp\\php886F.tmp", "updated_at": "2025-05-04T12:50:43.000000Z", "created_at": "2025-05-04T12:50:43.000000Z", "user_id": 22}}.

10. Pada Controllers/Api/RegisterController bagian create user ganti dengan

```
34 | 'image' => $request->image->hashName(),
```

11. Uji coba dan screenshot hasilnya apa perbedaan dari yang sebelumnya

Postman interface showing a successful POST request to <http://127.0.0.1:8000/api/register1>. The request body is form-data with fields: username (pengguna1ma), nama (pengguna5), password (12345), password_confirmation (12345), level_id (3), and image (icut.jpg). The response is JSON: {"success": true, "user": {"username": "pengguna1ma", "nama": "pengguna5", "level_id": 3, "image": "http://127.0.0.1:8000/storage/posts/OQPwPFNwJ14CqBREbUWukMvHuggaqrnTnizwN20.jpg", "updated_at": "2025-05-04T12:54:44.000000Z", "created_at": "2025-05-04T12:54:44.000000Z", "user_id": 23}}.

Perbedaan:

- Sebelum `"image": "http://127.0.0.1:8000/storage/posts/C:\\Users\\ASUS\\AppData\\Local\\Temp\\php886F.tmp",`
- Sesudah `"image": "http://127.0.0.1:8000/storage/posts/OQPwPFNwJ14CqBREbUWukMvHuggaqrnTnizwN20.jpg",`



TUGAS

Implementasikan API untuk upload file/gambar pada tabel lainnya yaitu tabel m_barang dan gunakan pada transaksi. Uji coba dengan method GET untuk memanggil data yang sudah di inputkan.

1. Implementasi Eloquent Accessor m_barang

- Membuat migration

```
C:\laragon\www\PWL_2025\Week11\JS11>php artisan make:migration add_image_to_m_barang_table  
INFO Migration [C:\laragon\www\PWL_2025\Week11\JS11\database\Migrations\2025_05_04_130820_add_image_to_m_barang_table.php]  
created successfully.
```

- Tambahkan kode pada migration yang sudah dibuat

```
JS11 > database > migrations > 2025_05_04_130820_add_image_to_m_barang_table.php > ...  
1  <?php  
2  
3  use Illuminate\Database\Migrations\Migration;  
4  use Illuminate\Database\Schema\Blueprint;  
5  use Illuminate\Support\Facades\Schema;  
6  
7  return new class extends Migration {  
8      public function up(): void  
9      {  
10         Schema::table('m_barang', callback: function (Blueprint $table): void {  
11             $table->string('image')->nullable(); // boleh null  
12         });  
13     }  
14  
15     public function down(): void  
16     {  
17         Schema::table('m_barang', callback: function (Blueprint $table): void {  
18             $table->dropColumn('image');  
19         });  
20     }  
21 };
```

- Jalankan migration

```
C:\laragon\www\PWL_2025\Week11\JS11>php artisan migrate  
INFO Running migrations.  
2025_05_04_130820_add_image_to_m_barang_table ..... 446ms DONE
```

- Tambahkan kode pada BarangModel.php

```
JS11 > app > Models > BarangModel.php > PHP Intellisense > BarangModel  
1  <?php  
2  
3  namespace App\Models;  
4  
5  use Illuminate\Database\Eloquent\Factories\HasFactory;  
6  use Illuminate\Database\Eloquent\Model;  
7  use Illuminate\Database\Eloquent\Relations\BelongsTo;  
8  use Illuminate\Database\Eloquent\Relations\BelongsToMany;  
9  use Tymon\JWTAuth\Contracts\JWTSubject;  
10  
11 30 references | 0 implementations  
12  class BarangModel extends Model  
13  {  
14      0 references | 0 overrides  
15      public function getJWTIdentifier(): mixed {  
16          return $this->getKey();  
17      }  
18  
19      0 references | 0 overrides  
20      public function getJWTClaims(): array { return []; }  
21  
22      use HasFactory;  
23  
24      0 references  
25      protected $table = 'm_barang';  
26      0 references  
27      protected $primaryKey = 'barang_id';  
28      0 references  
29      protected $fillable = [  
30          'kategori_id',  
31          'barang_kode',  
32          'barang_nama',  
33          'harga_jual',  
34          'image' // tambahkan  
35      ];  
36  
37      0 references | 0 overrides  
38      public function kategori(): BelongsTo  
39      {  
40          return $this->belongsTo(related: KategoriModel::class, foreignKey: 'kategori_id', ownerKey: 'kategori_id');  
41      }  
42  
43      0 references | 0 overrides  
44      public function stok(): HasOne  
45      {  
46          return $this->hasOne(related: StokModel::class, foreignKey: 'barang_id', localKey: 'barang_id');  
47      }  
48  
49      0 references | 0 overrides  
50      protected function image(): Attribute  
51      {  
52          return Attribute::make(  
53              get: fn ($image) => string::url($image ? $image : '/storage/barang/' . $image) : null  
54          );  
55      }  
56  }  
57  }
```



- Tambahkan pada Api\BarangController.php

```
117 public function destroy($id): JsonResponse|mixed
118 {
119     $barang = BarangModel::find(id: $id);
120
121     if (!$barang) {
122         return response()->json(data: [
123             'success' => false,
124             'message' => 'Barang tidak ditemukan',
125         ], status: 404);
126     }
127
128     $barang->delete();
129
130     return response()->json(data: [
131         'success' => true,
132         'message' => 'Barang berhasil dihapus',
133     ]);
134 }
135
```

- Hasil POST

http://127.0.0.1:8000/api/barangs

POST http://127.0.0.1:8000/api/barangs

Params Authorization Headers (8) Body Scripts Settings Cookies

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Key	Type	Value
harga_beli	Text	190000
harga_jual	Text	200000
image	File	
barang_id	Text	11
barang_nama	Text	Bostanten

Body Cookies Headers (10) Test Results

201 Created 834 ms 645 B

```
{
  "success": true,
  "data": {
    "kategori_id": "2",
    "barang_kode": "8011",
    "barang_nama": "Bostanten",
    "harga_beli": "190000",
    "harga_jual": "200000",
    "image": "http://127.0.0.1:8000/storage/barang/uwjh83wCYCKyL08eDjw068eP1HCH1A8GPa36T.jpg",
    "updated_at": "2025-06-04T13:31:03.000000Z",
    "created_at": "2025-06-04T13:31:03.000000Z",
    "barang_id": 14
  }
}
```

2. Implementasi Eloquent Accessor transaksi

- Membuat controller

C:\laragon\www\PWL_2025\Week11\JS11>php artisan make:controller Api\PenjualanController

INFO Controller [C:\laragon\www\PWL_2025\Week11\JS11\app\Http\Controllers\Api\PenjualanController.php] created successfully.

- Tambahkan kode pada PenjualanController

```
JS11 > app > Http > Controllers > Api > PenjualanController.php > PHP > PenjualanController
1 <?php
2
3 namespace App\Http\Controllers\Api;
4
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7 use App\Models\PenjualanModel;
8 use Illuminate\Support\Facades\Storage;
9 use Illuminate\Support\Facades\Validator;
10
11 class PenjualanController extends Controller
12 {
13     public function index(): JsonResponse|mixed
14     {
15         $data = PenjualanModel::with(relations: ['user', 'details.barang'])->get();
16         return response()->json(data: $data);
17     }
18
19     public function show($transaksi): JsonResponse|mixed
20     {
21         $penjualan = PenjualanModel::with(relations: ['user', 'details.barang'])->findOrFail(id: $transaksi);
22         return response()->json(data: $penjualan);
23     }
24 }
```




```
25 0 references | 0 overrides
26 public function store(Request $request): JsonResponse|mixed
27 {
28     $v = Validator::make($request->all(), rules: [
29         'user_id' => 'required|exists:m_user,user_id',
30         'pembeli' => 'required|string|max:100',
31         'penjualan_kode' => 'required|string|unique:t_penjualan,penjualan_kode',
32         'penjualan_tanggal' => 'required|date_format:Y-m-d H:i:s',
33         'image' => 'nullable|image|mimes:jpg,jpeg,png|max:2048',
34     ]);
35     if ($v->fails()) {
36         return response()->json(data: ['success'=>false,'errors'=>$v->errors()], status: 422);
37     }
38     $trx = PenjualanModel::create(attributes: $v->validated());
39
40     if ($request->hasFile(key: 'image')) {
41         $fn = time().'.'.$request->image->extension();
42         $request->image->storeAs('public/transaksi', $fn);
43         $trx->update(attributes: ['image'=>$fn]);
44     }
45
46     return response()->json(data: [
47         'success' => true,
48         'data' => [
49             'penjualan' => $trx,
50             'image_url' => $trx->image ? asset(path: 'storage/transaksi/'.$trx->image) : null,
51         ],
52     ], status: 201);
53 }
54
55 0 references | 0 overrides
56 public function update(Request $request, $transaksi): JsonResponse|mixed
57 {
58     $trx = PenjualanModel::findOrFail(id: $transaksi);
59
60     $v = Validator::make($request->all(), rules: [
61         'pembeli' => 'sometimes|required|string|max:100',
62         'penjualan_kode' => 'sometimes|required|string|unique:t_penjualan,penjualan_kode,$transaksi,penjualan_id',
63         'penjualan_tanggal' => 'sometimes|required|date',
64         'image' => 'nullable|image|mimes:jpg,jpeg,png|max:2048',
65     ]);
66
67     if ($v->fails()) {
68         return response()->json(data: ['success'=>false,'errors'=>$v->errors()], status: 422);
69     }
70
71     $trx->update(attributes: $v->validated());
72
73     if ($request->hasFile(key: 'image')) {
74         if ($trx->image) {
75             Storage::delete(paths: 'public/transaksi/'.$trx->image);
76         }
77         $fn = time().'.'.$request->image->extension();
78         $request->image->storeAs('public/transaksi', $fn);
79         $trx->update(attributes: ['image'=>$fn]);
80     }
81
82     return response()->json(data: $trx);
83 }
84
85 0 references | 0 overrides
86 public function destroy($transaksi): JsonResponse|mixed
87 {
88     $trx = PenjualanModel::findOrFail(id: $transaksi);
89     //cek file exists
90     if ($trx->image && Storage::exists(path: 'public/transaksi/'.$trx->image)) {
91         Storage::delete(paths: 'public/transaksi/'.$trx->image);
92     }
93     $trx->delete();
94     return response()->json(data: ['success'=>true,'message'=>'Transaksi dihapus']);
95 }
```

- Membuat migration

```
C:\laragon\www\PWL_2025\Week11\JS11>php artisan make:migration add_image_to_t_prenjualan_table
```

```
INFO Migration [C:\laragon\www\PWL_2025\Week11\JS11\database\Migrations\2025_05_04_134451_add_image_to_t_prenjualan_table.php]
created successfully.
```

- Tambahkan kode pada migration yang sudah dibuat

```
JS11 > database > migrations > 2025_05_04_134451_add_image_to_t_prenjualan_table.php > ...
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     public function up(): void
10     {
11         Schema::table('t_penjualan', callback: function (Blueprint $table): void {
12             $table->string(column: 'image')->nullable()->after(column: 'penjualan_tanggal');
13         });
14     }
15
16     public function down(): void
17     {
18         Schema::table('t_penjualan', callback: function (Blueprint $table): void {
19             $table->dropColumn(columns: 'image');
20         });
21     }
22
23 };
```




- Jalankan migration

```
C:\laragon\www\pwl_2025\Week11\JS11>php artisan migrate
INFO Running migrations.
2025_05_04_134451_add_image_to_t_prenjualan_table ..... 729ms DONE
```

- Tambahkan kode pada PenjualanModel.php

```
JS11 > app > Models > PenjualanModel.php > PHP Intelephense > PenjualanModel > user
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use Illuminate\Database\Eloquent\Relations\BelongsTo;
8 use Illuminate\Database\Eloquent\Relations\HasMany;
9 use App\Models\UserModel;
10 use App\Models\DetailPenjualanModel;
11 use Illuminate\Database\Eloquent\casts\Attribute;
12 use Tymon\JWTAuth\Contracts\JWTSubject;
13 use Illuminate\Foundation\Auth\User as Authenticatable;
14
15 class PenjualanModel extends Model
16 {
17
18     0 references | 0 overrides
19     public function getJWTIdentifier(): mixed{
20         return $this->getKey();
21     }
22
23     0 references | 0 overrides
24     public function getJWTCustomClaims(): array{
25         return [];
26     }
27 }
```

- Tambahkan kode pada routes api.php

```
12 use App\Http\Controllers\Api\PenjualanController;
60 // Route Transaksi
61 Route::get('transaksi', action: [PenjualanController::class, 'index']);
62 Route::get('transaksi/{transaksi}', action: [PenjualanController::class, 'show']);
63 Route::post('transaksi', action: [PenjualanController::class, 'store']);
64 Route::post('transaksi/{transaksi}', action: [PenjualanController::class, 'update']);
65 Route::delete('transaksi/{transaksi}', action: [PenjualanController::class, 'destroy']);
```

- Hasil POST

POST http://127.0.0.1:8000/api/transaksi?user_id=3& pembeli=Customer 11& penjualan_kode=TRX0011& penjualan_tanggal=2025-05-04 20 Save Share

POST http://127.0.0.1:8000/api/transaksi?user_id=3& pembeli=Customer 11& penjualan_kode=TRX0011& penjualan_tanggal=2025-05-04 20:57:00& image Send

Params Authorization Headers (8) Body Scripts Settings Cookies

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Key	Value
pembeli	Customer 11
penjualan_kode	TRX0011
penjualan_tanggal	2025-05-04 20:59:00
image	

Body Cookies Headers (10) Test Results Visualize

```
1 {
2   "success": true,
3   "data": {
4     "penjualan": {
5       "user_id": "3",
6       "pembeli": "Customer 11",
7       "penjualan_kode": "TRX0011",
8       "penjualan_tanggal": "2025-05-04 20:59:00",
9       "image": "http://127.0.0.1:8000/storage/transaksi/1746367192.jpg",
10      "updated_at": "2025-05-04T13:59:52.000000Z",
11      "created_at": "2025-05-04T13:59:51.000000Z",
12      "penjualan_id": 11
13    },
14    "image_url": "http://127.0.0.1:8000/storage/transaksi/http://127.0.0.1:8000/storage/transaksi/1746367192.jpg"
15  }
16 }
```

*** Sekian, dan selamat belajar ***