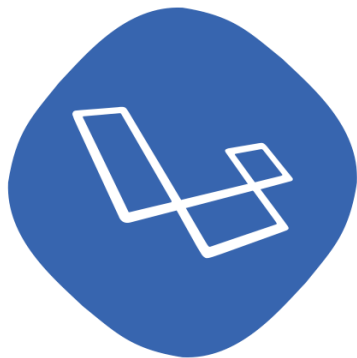


# لغات برمجة



# Laravel

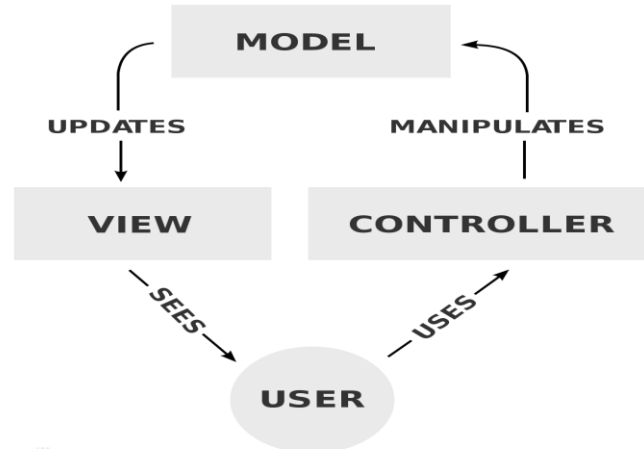
تقدمة الطالب :  
محمد بلال الشمالية

إشراف المهندسة :  
ريم الناصر

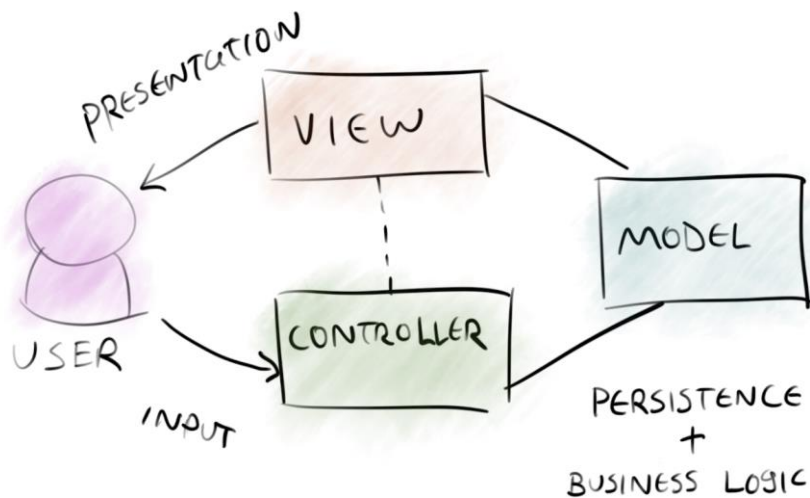
# ماهو الالارافيل :

الارافل هو اطار عمل خاص بلغة php , يقدم لارافل بيئة عمل متكاملة لكل ما تحتاجه لبناء موقع كامل من الناحيتين (back-end , front-end).

الارافل مبني على طريقة MVC الشهيرة التي تفصل ال Model عن View عن Controller , مما يوفر لك بيئة عمل سلسلة وسهلة.



# ما هو ال MVC ؟



◎ عمليات على قاعدة البيانات (إضافة - حذف - تعديل)  
مثال : إدخال معلومات طالب.

◎ استعراض البيانات.  
مثال : استعراض الطلاب.

## لماذا اللارافيل ؟

### واسع الانتشار

لأنها معتمدة على لغة برمجة مفتوحة المصدر .

### الامان

بسبب اتباعها خوارزميات تشفير من أجل تشفير البيانات المطلوب تشفيرها (كلمة المرور) بالإضافة لكونها مفتوحة المصدر ف إصلاح أخطاء الحماية يكون أسرع من قبل مطوريها

### Migration

وهو نظام تهجير البيانات حيث أن إنشاء قاعدة البيانات من خلاله أسهل من إنشائها يدويا بالإضافة لسهولة نقله لأنه أساسا موجود ضمن ملفات المشروع

بعد تنصيب برنامج xampp أو أحد البرامج الشبيه له , وبعد تنصيب الـ composer نضع الأمر التالي في الـ CMD .

⦿ composer create-project laravel/laravel example-app

⦿ بعد إنشاء المشروع يجب علينا تهيئة ملف الـ ENV الذي يهيئ التواصل مع قاعدة البيانات.  
⦿ بعدها يجب علينا إنشاء migration من أجل كتابة صفوف قاعدة البيانات بداخله وتهجير هذه الصفوف لتصبح قاعدة بيانات حقيقية.

تطبيق عملي : إنشاء صفحة لعرض وإدخال وحذف وتعديل بيانات طالب:

## Create Migration

Php artisan make:migrate  
student

## Create Model

Php artisan make:model  
Student

## Create Controller

Php artisan make:controller  
Student

## Create View

Throw create file with  
blade.php extinction

## ملف ال Migration الذي سيمثل جدول قاعدة البيانات :

```
public function up()
{
    Schema::create( table: 'students', function (Blueprint $table) {
        $table->id();
        $table->string( column: 'firstname');
        $table->string( column: 'lastname');
        $table->integer( column: 'age');
        $table->timestamps();
    });
}
```

```
/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists( table: 'students');
}
```

الطريقة up والتي ستستخدم من أجل إنشاء جدول في قاعدة البيانات حسب الأعمدة المذكورة في التاب create

الطريقة down والتي تستخدم من أجل حذف الجدول حذفاً نهائياً في حين استدعائها.

## طريقة تهجير ملف الـ Migration لكي يصبح جدولاً في قاعدة البيانات:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=stu
DB_USERNAME=root
DB_PASSWORD=
```

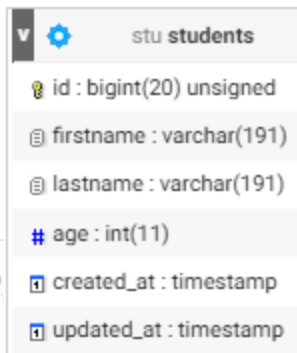
في البداية يجب تهيئة ملف الـ .env. الموجود في ملفات المشروع من أجل الاتصال مع قاعدة البيانات.

بعدها يجب كتابة الأمر :

php artisan migrate:fresh

```
Migrating: 2021_06_02_072431_create_students_table
Migrated: 2021_06_02_072431_create_students_table (246.01ms)

D:\My Project\students>
```



stu students
id : bigint(20) unsigned
firstname : varchar(191)
lastname : varchar(191)
# age : int(11)
created_at : timestamp
updated_at : timestamp

نلاحظ أنه تم تهجير الملف ليصبح جدول في قاعدة البيانات.

ملاحظة : يجب تشغيل الـ Xampp من أجل الـ phpMyAdmin



## صف الـ Model:

```
class Student extends Model
{
    use HasFactory;
    protected $fillable = ['firstname', 'lastname', 'age'];
}
```

وهو الصف الذي سيمثل أغراض الجدول student بحيث يكون كل سطر من البيانات المدخلة في الجدول عبارة عن غرض من الكلاس student .

وهو عبارة عن مصفوفة تأخذ القيم والتي هي عبارة عن أسماء الأعمدة في جدول الـ student .

## متحول الـ fillable:

سيتم الشرح عنها في سلايد لاحق 😊 .

## الـ HasFactory:

## الـ HasFactory:

والذي يدل على أن الـ Model لديه بيانات تجريبية من أجل الاختبار.

```
class StudentFactory extends Factory
{
    protected $model = Student::class;
    public function definition()
    {
        return [
            'firstname' => $this->faker->name,
            'lastname' => $this->faker->name,
            'age' => $this->faker->numberBetween( int1: 0, int2: 75)];
    }
}
```

من أجل ذلك يتم إنشاء صف يرث من الصف Factory وبداخله التابع definition يتم من خلاله إرجاع البيانات التجريبية كمصفوفة.

يتم إنشاء Factory من خلال الأمر:

```
php artisan make:factory StudentFactory --model="App\\Student"
```

يتم تهجير البيانات لقاعدة البيانات من خلال الأمر:

```
php artisan db:seed
```

## صف الـ Controller:

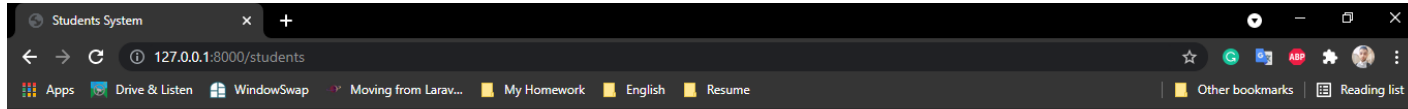
وهو الصف الذي ستنتم كتابه التوابع بداخله من أجل العمليات على قاعدة البيانات (استعراض - حذف - تعديل - إدخال).

## التابع Index:

وهو التابع المسؤول عن استدعاء بيانات الطلاب ومن ثم تمريرها للواجهة index التي سيتم من خلالها استعراض بيانات الطلاب.

```
public function index()
{
    $students = Student::orderBy( column: 'id', direction: 'desc')->get();
    return view( view: 'students.index',compact( var_name: 'students'));
}
```

المتحول student والذي سيحتوي على جميع بيانات الطلاب الموجودة في جدول الـ student.



## Students System

Create Student

No	Firstname	Lastname	Age	Action
20	Mr. Ike Von	Prof. Humberto Grady I	46	<a>Show</a> <a>Edit</a> <a>Delete</a>
19	Marcelle Reichel	Daphnee Keebler	32	<a>Show</a> <a>Edit</a> <a>Delete</a>
18	Samantha Rau	Hanna Hintz	44	<a>Show</a> <a>Edit</a> <a>Delete</a>
17	Winona Stark V	Gillian Towne	4	<a>Show</a> <a>Edit</a> <a>Delete</a>
16	Eudora Gaylord Sr.	Constantin Beahan	17	<a>Show</a> <a>Edit</a> <a>Delete</a>
15	Mr. Cordell Abshire	Dante Monahan	32	<a>Show</a> <a>Edit</a> <a>Delete</a>
14	Theo Bailey	Amelie Mills	31	<a>Show</a> <a>Edit</a> <a>Delete</a>
13	Miss Jewell Aufderhar DDS	Camille Barton	67	<a>Show</a> <a>Edit</a> <a>Delete</a>

## صفحة ال-Index:

الجزء المهم في هذه  
الصفحة هو حلقة

ال foreach حيث سيتم  
توليد سطر خاص بال-  
table من أجل كل  
طالب

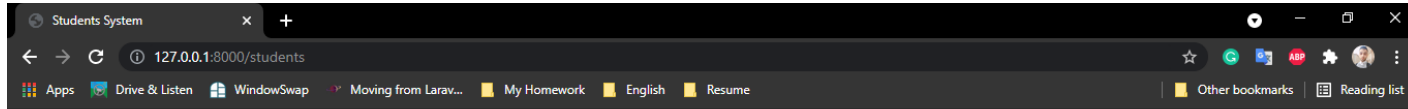
تنتهي الحلقة

بال-@endforeach  
anotation

```
<table class="table table-bordered">
  <tr>
    <th>No</th>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
    <th>Action</th>
  </tr>
  @foreach ($students as $student)
    <tr>
      <td>{{ $student->id }}</td>
      <td>{{ $student->firstname }}</td>
      <td>{{ $student->lastname }}</td>
      <td>{{ $student->age }}</td>
      <td><form action="{{ route('students.destroy',$student->id) }}" method="POST">
        <a class="btn btn-info" href="{{ route('students.show',$student->id) }}">Show</a>
        <a class="btn btn-primary" href="{{ route('students.edit',$student->id) }}">Edit</a>
        @csrf
        @method('DELETE')
        <button type="submit" onclick="return confirm('Do you really want to delete student!')" class="btn btn-danger">Delete</button>
      </form></td>
    </tr>
  @endforeach
</table>
```

نلاحظ وجود form وذلك من أجل عمليات الحذف و لتعديل و الاستعراض  
حيث يتم وصول للطالب من خلال رقمه (Id)

```
<td>
  <form action="{{ route('students.destroy',$student->id) }}" method="POST">
    <a class="btn btn-info" href="{{ route('students.show',$student->id) }}">Show</a>
    <a class="btn btn-primary" href="{{ route('students.edit',$student->id) }}">Edit</a>
    @csrf
    @method('DELETE')
    <button type="submit" onclick="return confirm('Do you really want to delete student!')" class="btn btn-danger">Delete</button>
  </form>
</td>
```



## Students System

Create Student

No	Firstname	Lastname	Age	Action
20	Mr. Ike Von	Prof. Humberto Grady I	46	<a>Show</a> <a>Edit</a> <a>Delete</a>
19	Marcelle Reichel	Daphnee Keebler	32	<a>Show</a> <a>Edit</a> <a>Delete</a>
18	Samantha Rau	Hanna Hintz	44	<a>Show</a> <a>Edit</a> <a>Delete</a>
17	Winona Stark V	Gillian Towne	4	<a>Show</a> <a>Edit</a> <a>Delete</a>
16	Eudora Gaylord Sr.	Constantin Beahan	17	<a>Show</a> <a>Edit</a> <a>Delete</a>
15	Mr. Cordell Abshire	Dante Monahan	32	<a>Show</a> <a>Edit</a> <a>Delete</a>
14	Theo Bailey	Amelie Mills	31	<a>Show</a> <a>Edit</a> <a>Delete</a>
13	Miss Jewell Aufderhar DDS	Camille Barton	67	<a>Show</a> <a>Edit</a> <a>Delete</a>

التابع create:

وهو التابع المسؤول عن استعراض صفحة إضافة طالب, ويتم استدعائه عند الضغط على زر create student في واجهة ال-Index .

```
public function create()
{
    return view( view: 'students.create');
}
```

## Create Student

**Firstname:**

**Lastname:**

**Age:**

Back

Submit



## التابع store:

وهو التابع المسؤول عن إدخال البيانات الموجودة ضمن الـ request والتي من المفترض أن تكون بيانات طالب.

يتم التحقق من وجود البيانات من خلال التابع validate ويتم إدخالها عن طريق التابع create بعد ذلك يتم توجيهه للصفحة index .

```
public function store(Request $request)
{
    $request->validate([
        'firstname' => 'required',
        'lastname' => 'required',
        'age' => 'required',
    ]);

    Student::create($request->all());
    return redirect()->route( route: 'students.index')->with('success', 'Student created successfully.');
```

## Create Student

Firstname:

Belal

Lastname:

Al Shmalya

Age:

23

Back

Submit

Student created successfully.

No	Firstname	Lastname	Age	Action
21	Belal	Al Shmalya	23	<a>Show</a> <a>Edit</a> <a>Delete</a>

## صفحة إنشاء طالب:

```
<form action="{{ route('students.store') }}" method="POST">
  @csrf
  <div class="row">
    <div class="col-xs-12 col-sm-12 col-md-12">...</div>
    <div class="col-xs-12 col-sm-12 col-md-12">...</div>
    <div class="col-xs-12 col-sm-12 col-md-12">...</div>
    <div class="col-xs-12 col-sm-12 col-md-12 text-center">
      <a class="btn btn-primary" href="{{ route('students.index') }}"> Back</a>
      <button type="submit" class="btn btn-success">Submit</button>
    </div>
  </div>
</form>
```

الجزء المهم في هذه الصفحة هو ال form والذي سيتم من خلاله إرسال البيانات للتابع create عند الضغط على الزر submit

## التابع edit:

وهو التابع المسؤول عن استعراض صفحة تعديل طالب, ويتم استدعائه عند الضغط على زر Edit في واجهة ال-Index بحيث يتم الاستعلام عن بيانات الطالب من خلال ال-id الخاص به الذي تم تمريره من الصفحة index .

```
public function edit(Student $student)
{
    return view( view: 'students.edit', compact( var_name: 'student' ));
}
```

### Edit Student

Firstname:

Lastname:

Age:

[Back](#)[Update](#)

## التابع update:

وهو التابع المسؤول عن إدخال البيانات الموجودة ضمن الـ request والتي من المفترض أن تكون بيانات طالب.

يتم التحقق من وجود البيانات من خلال التابع validate ويتم الاستعلام عن الطالب عن طريق التابع update ومن ثم تعديل بيانات الطالب بحسب المدخلة من الـ request بعد ذلك يتم توجيهه للصفحة index .

```
public function update(Request $request, Student $student)
{
    $request->validate([
        'firstname' => 'required',
        'lastname' => 'required',
        'age' => 'required',
    ]);
    $student->update($request->all());
    return redirect()->route( route: 'students.index')->with('success', 'Student updated successfully');
}
```

## صفحة تعديل طالب:

```
<form action="{{ route('students.update', $student->id) }}" method="POST">
    @csrf
    @method('PUT')
    <div class="row">
        <div class="col-xs-12 col-sm-12 col-md-12" ...>
        <div class="col-xs-12 col-sm-12 col-md-12" ...>
        <div class="col-xs-12 col-sm-12 col-md-12" ...>
        <div class="col-xs-12 col-sm-12 col-md-12 text-center">
            <a class="btn btn-primary" href="{{ route('students.index') }}"> Back</a>
            <button type="submit" class="btn btn-success">Update</button>
        </div>
    </div>
</form>
```

الجزء المهم في هذه الصفحة هو ال form والذي سيتم من خلاله إرسال البيانات مع ال id الخاص بالطالب للتابع update عند الضغط على الزر submit

## Edit Student

Firstname:

Mohamad Belal

Lastname:

Al-Shmalya

Age:

22

Back

Update



Student updated successfully

No	Firstname	Lastname	Age	Action
21	Mohamad Belal	Al-Shmalya	22	<div>Show</div> <div>Edit</div> <div>Delete</div>

## التابع `destroy`:

وهو التابع المسؤول حذف طالب ويتم استدعائه عند الضغط على الزر `delete` في الصفحة `.index`.

```
public function destroy(Student $student)
{
    $student->delete();
    return redirect()->route( route: 'students.index')->with('success', 'Student deleted successfully');
}
```



زر حذف طالب:

Create Student

Do you really want to delete student!

OK

Cancel

No	Firstname	Lastname	Age	Action
21	Mohamad Belal	Al-Shmalya	22	<a>Show</a> <a>Edit</a> <a>Delete</a>

Students System

Create Student

Student deleted successfully

## ملف ال route < web.php:

وهو الملف المسؤول عن توجيه المتصفح لاستدعاء التوابع المذكورة سابقا  
وبما انه تم اعتماد المقياس العالمي CRUD فلا يوجد حاجة لكتابة نوع ال request فقط  
نحتاج كتابة السطر الموجود أدناه.

```
/*
|-----
| Web Routes
|-----
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::resource( name: 'students', controller: \App\Http\Controllers\StudentController::class);
```

## المراجع المستخدمة:

- © <https://laravel.com/>
- © <https://www.sumologic.com/glossary/crud/>
- © <https://techterms.com/definition/mvc>

## رابط المشروع على الـ Github:

- © <https://github.com/belaloo/Programing-Language-Project.git>

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are solid grey and others are hollow with a grey outline. The lines are thin and grey, connecting the nodes in a non-linear fashion. The overall shape of the network is roughly triangular, pointing towards the top-left corner of the slide.

# Thank You.

M Belal Al-Shmalya

A decorative network diagram in the bottom-right corner, similar to the one in the top-left. It consists of a cluster of nodes (solid grey circles and hollow circles with grey outlines) connected by thin grey lines. The network is more spread out and less dense than the one in the top-left, with some nodes having multiple connections and others being isolated.