التعامل مع قواعد البيانات في لارافل

السلام عليكم و رحمة الله و بركاته
في هذه المحاضرة ستتعرف على طريقة التعامل مع قواعد البيانات في لارافل ، و كذلك سنتطرق إلى مفهوم تهجير قواعد البيانات (Migration) و الغرض من استخدامه .

لارافل تدعم حالياً أربع أنواع من قواعد البيانات :

MySQL

PostgreSQL

SQLite

SQL Server

و نوع قاعدة البيانات التي سأستخدمها هنا للتطبيق ستكون MySQL

إعدادات الإتصال بقاعدة البيانات
لبدء استخدام قواعد البيانات ينبغي ضبط الإعدادات الخاصة بالاتصال بقاعدة البيانات ، الإعدادات الخاصة بالتعامل مع قواعد البيانات موجودة بداخل الملف **database.php** و الموجود بداخل المجلد **config** ، و يتم تحديد القيم لهذه الإعدادات من خلال الملف **env**. الموجود في الجذر الرئيسي لمشروع لارافيل .
من خلال ملف env يُحدد نوع قاعدة البيانات التي ستتعامل معها ، أيضاً اسم قاعدة البيانات مع اسم المستخدم و كلمة المرور.

في لارافل يمكن التعامل مع قواعد البيانات بثلاث طرق مختلفة :
Eloquent ORM , Query builder , Raw SQL
و في هذه المحاضرة سنركز على النوعين Query builder و Raw SQL بينما ستُخصص المحاضرة القادمة للطريقة الثالثة Eloquent ORM .

Raw SQL Queries

الطريقة الأولى للتعامل مع قاعدة البيانات هي Raw SQL ، في هذا النوع يتم التعامل مع قاعدة البيانات بشكل مباشر باستخدام جمل SQL المعتادة ، و يفضل استخدام هذه الطريقة في حالة تنفيذ استعلامات شديدة التعقيد على قاعدة البيانات .

لتنفيذ استعلامات Raw SQL تستخدم الواجهة DB

فمثلاً لجلب بيانات حسب شرط معين سنكتب التالي
، نبدأ بـDB و بما أنه استعلام لاسترداد بيانات فإذاً سنستخدم Select ، و بين الأقواس تُكتب جملة sql ، و هنا سيتم جلب بيانات الـuser الذي رقمه Id هو ١ .

```
$result = DB::select('select * from users where id=:id', [1]);
```

و لإدخال بيانات سنستخدم insert بهذه الطريقة

```
DB::insert('insert into users(id,name) values (?, ?)', [5, 'Ali']);
```

تحديث سجلات معينة

```
$affected = DB::update('update users set name = "sarah" where id = ?', [1]);
```

حذف بيانات

```
$deleted = DB::delete('delete from users');
```

سنقوم الآن بعمل مثال بسيط لعرض جميع بيانات المستخدمين من جدول users
بما أنه سنتعامل مع قاعدة بيانات ، إذاً لا بد أولاً من تحديد اسم قاعدة البيانات مع اسم المستخدم و كلمة المرور من خلال الملف env ، اسم قاعدة البيانات لدي هي **laravel_db** و اسم المستخدم هو **root** و لم أعين باسوورد لها فإذاً ستكون القيمة هنا **NULL** .

سنبدأ بإنشاء الـview الذي ستعرض عليه البيانات، و سأقوم بتسميته users ، البيانات التي سيتم جلبها من قاعدة البيانات ستعرض على هذا الـview

أريد عرض هذا الـView على المتصفح عند طلب العنوان **users/show** ، إذاً سأنتقل للملف الخاص بإنشاء المسارات و هو web.php ، و أكتب التالي :

```
Route::get('/users/show', function () {
    return view('users');
});
```

جيد . تم عرض الـ View المسمى **users** عند طلب العنوان **/users/show** ، و لكن كما ترى فالصفحة فارغة ، و ما نريده هو جلب البيانات من الجدول users و عرضها هنا ، إذاً سنستخدم التعليمات الخاصة بالتعامل مع قاعدة

البيانات و كما ذكرنا سابقاً فإن هذه التعليمات تكتب بداخل متحكم Controller و هو بدوره سيقوم بجلبها من قاعدة البيانات ثم إرسالها لـView .

سننشيء متحكم بالإسم UserController ، و سأضيف بداخله function جديد بالاسم show ، ستكون مهمة هذا الـfunction هو جلب جميع البيانات من جدول users ثم إرسال النتائج إلى الـview المسمى Users .

إذاً ستكتب الجملة بداخل show() بهذا الشكل :

```
$result = DB::select('select * from users ');
```

ثم سيرسل النتيجة الموجودة في $result إلى ملف العرض بشكل مصفوفة بالطريقة

```
return view('users', array('results' => $result));
```

أو يمكن عمل هذا بشكل مختصر باستخدام Compact ، سيصبح السطر بهذا الشكل و هو يماثل السابق

```
return view('users', compact('results'));
```

و الآن قمنا بتمرير البيانات القادمة من قاعدة البيانات إلى الـ view المسمى users .

فلننتقل الآن للـview (users) و نقوم بعرض البيانات ، سنستخدم حلقة foreach للمرور على جميع العناصر الموجودة في results و طباعتها .

```
@if(isset($results))
     @foreach($results as $rslt)
           {{$rslt->id}}   <b>
           {{$rslt->name}} <br>
           {{$rslt->email}}<br>
     @endforeach
@endif
```

إذا قمنا الآن بطلب العنوان **/users/show** سيعرض لنا الفيو users و لكن بدون البيانات .**فما السبب ؟**

إذا لاحظت هنا للمسار في ملف web ستجد أنه عند طلب العنوان **/users/show** سيتوجه للفيو users مباشرةً ، أي دون المرور على الـController ، **إذاً** سنعمل هنا على تعديل المسار ليقوم بالتوجه إلى الكونترولر المسئول عن طلب بيانات الـusers ، و هو هنا الـUsersController و اسم الـfunction التي ستقوم بالمهمة هي show

```
Route::get('users/show','UserController@show')->name('show');
```

(الجملة name->('show') مجرد تسمية لهذا المسار ، سنستخدمه لاحقاً)

الآن جميع الأمور جاهزة .

فلنقم بطلب العنوان **/users/show** مرة أخرى و نشاهد النتيجة ،

تم عرض البيانات بنجاح

الآن يمكنك ترتيب طريقة عرض البيانات ، فلنقم مثلاً بعرضها بشكل جدول .

```
<table>
    <tr>
        <th>id</th>
        <th>name</th>
        <th>email</th>
    </tr>
    @foreach($results as $rslt)
        <tr>
            <td>{{$rslt->id}} </td>
            <td>{{$rslt->name}} </td>
            <td>{{$rslt->email}}</td>
        </tr>
    @endforeach
</table>
```

كما رأيت فـ Raw SQL تستخدم استعلامات SQL الإعتيادية ، و من المؤكد أنك تعاملت معها مسبقاً ، يمكنك تنفيذ إستعلامات من عدة جداول و مهما كان درجة تعقيدها باستخدام هذه الطريقة .

Query builder

طريقة أخرى من طرق التعامل مع قواعد البيانات في لارافل و هي Query builder ، تتمثل ميزة استخدام Query builder في تبسيط طرق كتابة جمل SQL ، تعمل Query builder على جميع أنواع قواعد البيانات التي تدعمها لارافل ، و تستخدم عوامل من شأنها أن تحمي تطبيق الويب ضد هجمات SQL injection .

لنرى الآن طريقة جلب جميع البيانات في جدول معين باستخدام Query builder

تبدأ إستعلامات Query builder بالواجهة DB ثم الدالة table ، يذكر بعدها اسم الجدول الذي سنستعلم منه ، و في حالة أن المطلوب هو جلب جميع البيانات بدون أي شروط إذاً ستكتب get مباشرةً . هذه الجملة ستعطينا جميع البيانات في جدول الـ users ، و الدالة get هنا ستعيد النتائج بشكل Collection .

```
$users = DB::table('users')->get();
```

ماذا إذا احتجنا لبيانات مستخدم معين ، فمثلاً أريد جلب بيانات المستخدم الذي اسمه Mohammed ، سنضيف شرط where بهذا الشكل بحيث يذكر اسم الحقل الذي سنستعلم بواسطته و هو هنا الحقل name و قيمته هو mohammed .

```
$user = DB::table('users')
        ->where('name', 'Mohammed')
        ->first();
```

استخدمنا الدالة first لتعيد سجل واحد فقط ، و هو هنا أول سجل يحقق الشرط 'name = 'mohammed.

يمكن كذلك تحديد الحقول التي نحتاجها فقط ، عوضاً عن جلب بيانات جميع الحقول بالشكل التالي :

```
    $users = DB::table('users')
            ->where('name', 'Mohammed')
            ->select('name','email')
            ->get();
```

هنا اخترنا فقط الحقلين name و email من أصل ثلاثة حقول في الجدول .

يمكن أيضاً عن طريق Query builder الإستعلام من أكثر من جدول عن طريق join ، و تستخدم بهذا الشكل :

```
$users = DB::table('users')
      ->join('contacts', 'users.id', '=', 'contacts.user_id')
      ->join('orders', 'users.id', '=', 'orders.user_id')
      ->select('users.*', 'contacts.phone', 'orders.price')
      ->get()
```

و هذا الإستعلام يجلب بيانات من ثلاث جداول users و contacts و orders ، الوسيطة الأولى هي الجدول الأساسي ، ثم تستخدم join لذكر الجداول الأخرى المرتبطة ، مع شرط الربط مع الجدول الأساسي ، و هنا select لتحديد أسماء الحقول المطلوب جلبها من كل جدول.

يمكنك تنفيذ مختلف أنواع الإستعلامات على قواعد البيانات باستخدام Query builder ، و لا يتسع المجال لسرد جميعها هنا ، يمكنك الإطلاع على كافة الطرق الممكنة لاستخدام Quiry builder من خلال التوثيق الرسمي للارفيل .

https://laravel.com/docs/5.5/queries

سنقوم الآن بعمل مثال بسيط لتوضيح عملية عرض و حذف البيانات بـ Query builder

لدينا هنا نفس الكونترولر الذي طبقنا عليه االمثال السابق ، نريد فقط تغيير هذا الإستعلام ، بحيث نقوم بجلب بيانات المستخدمين بإستخدام Query builder عوضاً عن الطريقة المباشرة ، إذا سنكتب الجملة بهذا الشكل

```
$users = DB::table('users')->get();
```

ثم سنمرر البيانات إلى الـ view المسمى users ، و في الـview ستعرض البيانات بشكل جدول ، فلنقم بالتحقق من النتائج

و الآن بجانب كل صف من هذه الصفوف سنضيف رابط للحذف ، بحيث يؤدي عند النقر عليه إلى حذف السجل من قاعدة البيانات .

فلنعد للـView و نضيف رابط تشعبي كعنصر جديد ضمن الجدول

عند النقر على هذا الرابط سيتم تنفيذ مهمة معينة و هي عملية الحذف ، إذاً علينا توجيه الطلب إلى Function بداخل كونترولر ليقوم بمهمة الحذف .

سنبدأ بإنشاء الـFunction الخاص بالحذف بداخل الـUserController ، و سأعطيه الإسم delete ، هذا الـFunction سيستقبل رقم السجل المطلوب حذفه ، سيُكتب أمر الحذف بالشكل التالي:

```
        DB::table('users')
            ->where('id', '=', $id)
            ->delete();
```

تستقبل الدالة table اسم الجدول ، نحدد شرط الحذف و هو هنا (عندما يكون قيمة الحقل id مساوياً للرقم المرسل) ، ثم أخيراً الإجراء الذي سينفذ عند تطابق الشرط و هو الحذف delete.

بقي لنا استدعاء هذه المهمة لتنفيذها عند النقر على الإرتباط التشعبي **delete** ، فلنحدد عنوان معين و ليكن users/delete/ و لاحظ هنا أنه يجب تمرير قيمة الـuser ID و الممثل بـ**$rslt ->id**

```
<a href="/users/delete/{{$rslt->id}}"> delete </a>
```

ماذا نعني بهذا السطر؟

إذا قمنا الآن بالنقر على الرابط delete ، سيقوم مباشرة بطلب العنوان الذي أنشأناه و هو users/delete/ مع الـId للسجل المحدد ، و لكن هذا العنوان غير معروف ، لأنه غير مسجل في ملف الـRoutes و لهذا السبب تظهر صفحة الخطأ ، فلنقم الآن بتسجيل هذا العنوان في ملف web.php بحيث يستدعي دالة الحذف عند طلبه

```
Route::get('/users/delete/{id}', 'UserController@delete');
```

معنى هذا السطر هو قم باستدعاء المهمة delete و الموجودة بداخل الـController المسمى UserController ، عندما يتم طلب العنوان users/delete متبوعاً بأي قيمة كانت .

فلنقم الآن بمشاهدة النتائج ، تمت عملية الحذف ، و لكن لاحظ أنه يقوم بالإنتقال إلى صفحة أخرى بعد عملية الحذف ، لذا سأقوم بإضافه بسيطة هنا ليتوجه للعنوان users/show بعد كل عملية الحذف ، و هنا سأستخدم إسم الـRoute و هو show .

```
return redirect()->route('show');
```

ليتضح لك هذا السطر ، دعنا ننتقل للملف الخاص بالمسارات ، لاحظ هنا أن المسار الذي يؤدي لصفحة بيانات الـusers اسمه (name) هو show ، و ما قمنا به هو استخدام هذا الاسم في التوجيه بدلاً عن عنوانه .

فلنجرب مرة أخرى بعد التعديل ، تمت عملية الحذف مع إعادة التوجيه لنفس الصفحة .

و الآن قم بتطبيق جميع ما سبق مع إضافة أمر التعديل بنفس الطريقة .

تهجير قواعد البيانات
Migration

لارافل توفر طريقة لإدارة قواعد البيانات بإستخدام ما يسمى بالتهجير أو Migration
ففي الوضع العادي ستقوم بإنشاء الجداول بالطريقة اليدوية المعتادة ، و في حالة ما أردت حذف أو تعديل حقول قاعدة البيانات فسيتم هذا بطريقة مملة و مرهقة ، كما لن يمكنك تنفيذ أمر التراجع عن أي عملية أجريتها على قاعدة البيانات .

الـMigration هو آلية تسهل من عملية إنشاء و تعديل جداول قاعدة البيانات ، بحيث يتم وصف الجداول و الحقول بداخل ملفات ضمن مشروع لارافل ، ثم تنفيذ الأمر **migrate** ليتم ترحيل هذه الحقول إلى قاعدة البيانات ، و عند الرغبة في التعديل على هذه الجداول أو الحقول فسيتم التعديل على هذه الملفات بطرق سهلة ، و يمكنك إعادة إنشاء هذه الجداول في أي وقت و على أي قاعدة بيانات بطريقة سهلة.

توجد ملفات الـ Migration بداخل المجلد `database/migrations` و يمثل كل ملف من هذه الملفات جدول واحد ، يتواجد ملفي Migration افتراضيين ضمن مشروع لارافل هما users و password_resets ، و يمكنك إضافة أخرى حسب الجداول التي ستضيفها لقاعدة البيانات .

لنفترض أننا سنضيف جدول بالإسم products لقاعدة البيانات ، إذاً علينا أولاً إنشاء ملف الـ Migration هنا و ستقوم بإنشاءه عن طريق موجه الأوامر بواسطة الأمر

```
php artisan make:migration create_ products _table --create= products
```

الجملة table_ create_ products هي إسم اختياري للملف (و يفضل تسمية ملفات التهجير بنفس هذه الطريقة)
و products هو اسم الجدول الذي سنتعامل معه
فللنتقل لملف الـMigration الذي قمنا بإنشاءه ، ستلاحظ وجود دالة up و هي لبناء الجدول في قاعدة البيانات ، و الدالة down و هي ما سينفذ عند التراجع عن عملية التهجير rollback.
و الآن ضمن الدالة up سنقوم بوصف الحقول التي نحتاجها ، و ستلاحظ وجود وصف لحقلين
id و هو معرف للجدول (مفتاح أساسي)
و **timestamp** و هو ينشيء الحقلين created_at و updated_at في قاعدة البيانات
سنضيف أيضاً حقل من نوع string بالاسم product_name

والآن فلنقم بعملية تهجير هذه الجداول users و password_resets و products إلى قاعدة البيانات ، و سيتم هذا باستخدام الأمر **php artisan migrate** ، مع العلم أن عملية التهجير ستكون إلى نفس قاعدة البيانات التي حددنا إسمها داخل الملف env .

فلننفذ أمر migrate لإنشاء الجداول في قاعدة البيانات .

```
C:\Users\R_lotf\Desktop\laravelProject>php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table
Migrating: 2018_02_13_202647_create_table_products
Migrated:  2018_02_13_202647_create_table_products
Migrating: 2018_02_13_211037_create_table_products_details
Migrated:  2018_02_13_211037_create_table_products_details
```

لاحظ أنه تم إنشاء الجداول على قاعدة البيانات فعلاً ، كرر نفس الخطوات مع أي جدول جديد ترغب بإنشائه

سأضيف مثلاً جدول بالاسم **Product_details**

```
php artisan make:migration create_products_details_table --create=products_details
```

و سأنفذ مرة أخرى الأمر **migrate** لينشئ الجدول الجديد في قاعدة البيانات .

```
C:\Users\R_lotf\Desktop\laravelProject>php artisan migrate
Migrating: 2018_02_14_065450_create_products_details_table
Migrated:  2018_02_14_065450_create_products_details_table
```

لاحظ أنه نفذ عملية التهجير للجدول الجديد فقط .

ماذا إذا رغبنا بإجراء تعديلات على هذا الجدول ، مثلاً سأقوم بتنفيذ تعديل بسيط ، سأضيف حقل جديد هنا ، و أرغب أيضاً في تعيينه كـforeign key

لتعيين الحقل كـforeign key يجب تحديده أولاً كـUNSIGNED حتى يتوافق مع المفتاح الرئيسي و الذي يحمل نفس هذه الخاصية

الخطوة الثانية هي تعيينه كـforeign ثم سنحدد إلى ماذا سيشير هذا الحقل ، و بالطبع سيشير إلى الحقل id الموجود على الجدول products .

```
$table->integer('product_id')->unsigned();
$table->foreign('product_id')->references('id')->on('products');
```

و الآن ينبغي تطبيق هذه التغييرات على قاعدة البيانات ، و إذا قمت بتنفيذ الأمر php artisan migrate فستظهر الرسالة nothing to migrate لأنه بالفعل ليس هناك ملفات تهجير جديدة أضيفت هنا ليتم ترحيلها إلى قاعدة البيانات ، فكيف إذاً سيطبق التغييرات الجديدة على جدول تم إنشاءه فعلياً ؟

الحل هو باستخدام أحد الأوامر refresh أو fresh ، و هما متشابهتان إلى حد كبير

فالـRefresh ستقوم بالتراجع عن عملية الـmigration المنفذة سابقاً و بعدها تعيد تطبيقها من جديد

أما **Fresh** ستعمل على حذف جميع الجداول في قاعدة البيانات ثم إعادة إنشائها .

سأنفذ الأمر **php artisan migrate : refresh**

دعنا نلقي نظرة على قاعدة البيانات لنشاهد الجداول التي تم إنشاءها

```
mysql -u username –p password
show tables;
```

هذه جميع الجداول التي تم إنشاءها على قاعدة البيانات ، فلنستعرض مثلاً الجدول product_details لتتأكد من نوع الحقول

```
> Desc product_details;
```

كما ترى هذه جميع الحقول التي أضفناها في ملف التهجير و ستلاحظ أن timestamp ينشيء حقلين created_at و updates_at .

كما و يمكنك التراجع عن عملية Migration التي أجريتها باستخدام الأمر rollback بهذا الشكل

```
php artisan migrate:rollback
```

إذا كنت تستخدم إصدار من mysql أقدم من ٥.٧.٧ أو MariaDB أقدم من الإصدار ١٠.٢.٢ فقد تظهر لديك الرسالة التالية عند تنفيذ عملية التهجير Migration

```
SQLSTATE[42000]: Syntax error or access violation: 1071 Specified key was too
long; max key length is 767 bytes
```

لحل هذه المشكلة قم بتحديد الطول الإفتراضي للسلسلة بإضافة التعليمة البرمجية التالية في الملف **AppServiceProvider.php** الموجود على الدليل **App/providers**

```
public function boot()
{
    Schema::defaultStringLength(191);
}
```

و لا تنسى تضمين الواجهة schema في الأعلى بهذا الشكل

```
use Illuminate\Support\Facades\Schema;
```

يمكنك زيارة الرابط التالي للتعرف على المزيد حول الـMigration

https://laravel.com/docs/5.5/migrations