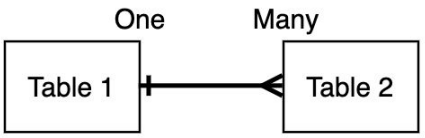
This Cheat-Sheet belongs to the Laravel-Beginner-Course on Udemy:
https://www.udemy.com/course/laravel-7-for-beginners-practical-course

Laravel Database relationships + seeding - Cheat Sheet

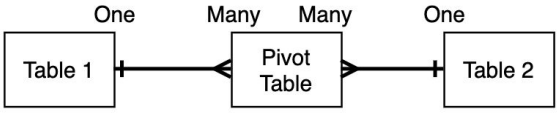
One-To-Many-Relationship



Model 1: $this->**hasMany**('Model 2');

Model 2: $this->**belongsTo**('Model 1');

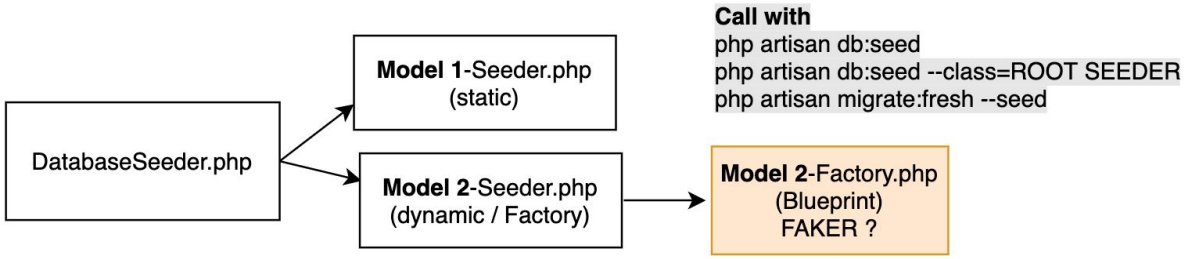
Many-To-Many-Relationship



Model 1: $this->**belongsToMany**('Model 2');

Model 1: $this->**belongsToMany**('Model 1');

SEEDER / FACTORY / FAKER



Call other Seeder from DatabaseSeeder

```
public function run()
{
   $this->call(OtherSeeder::class);
   $this->call(UserSeeder::class);
}
```

Example Factory (= Blueprint)

```
use App\Hobby;
use Faker\Generator as Faker;

$factory->define(Hobby::class, function (Faker
$faker) {
   return [
      'name' => $faker->realText(30),
      'description' => $faker->realText(),
```

Migration Example

```
public function up()
{
   Schema::table('hobbies', function (Blueprint $table)
{
      $table->unsignedBigInteger('user_id')
      ->after('id')
      ->nullable();
      $table->foreign('user_id')
         ->references('id')->on('users')
         ->onDelete('cascade');
   });
}

public function down()
{
   Schema::table('hobbies', function (Blueprint $table)
{
      $table->dropForeign(['user_id']);
      $table->dropColumn('user_id');
   });
}
```

This Cheat-Sheet belongs to the Laravel-Beginner-Course on Udemy:
https://www.udemy.com/course/laravel-7-for-beginners-practical-course

Changes to databases

- When you are already online: Make new migration, eg. "**change_xyz-table**" and run `php artisan migrate` again.
- When still in local development you can use
  `php artisan migrate:fresh` or `php artisan migrate:fresh --seed`
- With `php artisan migrate:rollback` you can go back one step in your migrations history

Laravel Tinker

You can use Laravel Tinker to simulate Laravel Code without Frontend.
You start Tinker with `php artisan tinker` und you end Tinker with `exit`

Example Usage of Tinker:

- `App\Hobby::first()` - Get first instance of Hobby Model
- `App\Hobby::first()->tags` - Get all the Tags for the first Hobby Model instance
- `App\Hobby::first()->tags->pluck(name)` - Get all the Tags for the first Hobby Model instance, but only show the name field of the tags.

Working with Many-To-Many relationships in practice

Filtering

When you have a Many-To-Many relationship, e.g. 'products' and its 'categories' you most likely want to filter your products by category. You can do this by defining a new relationship in the 'category' -Model:

```
public function filteredProducts() {
  return $this->belongsToMany('App\Product')
    ->wherePivot('catogory_id', $this->id)
}
```

Then in any controller, that imports the Category Model you can get the filtered Products by eg:

```
$category = new Category();
$filteredProducts = $category::findOrFail($category_id)->filteredProducts()
```

Assigning / Deleting single Many-To-Many relationships between two Models

*The **attach()** method and the **detach()** method:*
For the example of 'products' and its 'categories' you need a controller where you import both - the Product-Model and (probably) the Category-Model. Then you can say:

```
public function attachCategory($product_id, $category_id) {
  $product = Product::find($product_id);
  $product->categories()->attach($category_id);
  ….
```

The opposite you can do if you use the **detach()** method instead of the **attach()** method.