wemwgjøvwni ivngvwbi ivwng

Follow on: https://www.youtube.com/watch?v=L6xaLfuqGfU&list=PLjsp2uDzfb31NqnycsDO41j18bYeKOGp-&index=4&ab\_channel=WebJourney  
jviv‡fj ïiæ:

1. composer create-project --prefer-dist laravel/laravel blog

Zvici

1. php artisan serve

KgvÛ w`‡q †`Le mvfv©i ivb K‡i wKbv Ges[**http://127.0.0.1:8000/**](http://127.0.0.1:8000/) GLv‡b wM‡q †`Le mvBU wVKg‡Zv BÝUªj n‡q‡Q wKbv | wVK \_vK‡j mvBU †`Lv‡e |  
Gevi mivmwi P‡j hv‡ev ‡KvW GwWU‡i Ges cª‡Rw± I‡cb Kie |

Gici .env dvBj I‡cb K‡i WvUv‡e‡mi bvg BDRvi bvg I cvmIqvW© ewm‡q †mf w`‡ev |

1. DB\_CONNECTION=mysql
2. DB\_HOST=127.0.0.1
3. DB\_PORT=3306
4. DB\_DATABASE=laravel
5. DB\_USERNAME=root
6. DB\_PASSWORD= 123

jviv‡fj †RUw÷ªg A‡\_bwU‡Kkb wm‡÷g Pvjy Kivi Rb¨ wb‡Pi avc¸‡jv Abymib Kiæb | wb‡Pi KgvÛ ivb Kiæb

1. composer **require** laravel/jetstream

Gevi †RUw÷ªg Gi mv‡\_ jvBfIq¨vi BÝUªj Kivi Rb¨ wb‡Pi KgvÛ ivb Kiæb |

1. php artisan jetstream:install livewire
2. php artisan jetstream:install livewire --teams
4. #Or, Install Jetstream With Inertia
6. php artisan jetstream:install inertia
7. php artisan jetstream:install inertia --teams

Gevi wb‡Pi KgvÛ¸‡jv ivb Kiæb

1. npm install
2. npm run dev

jviv‡fj †mUAvc †kl | Gevi Avgiv w¯‹gv wjLvi wbqg wb‡q KvR Kie | w¯‹gv wjLvi A‡bK wbqg Av‡Q Avgiv wewfbœfv‡e wjL‡Z cvwi | Avmyb Zvn‡j ‡`wL KZ iKgfv‡e w¯‹gv wjLv hvq |

GK †Uwe‡ji AvBwW hw` Avgiv Ab¨ †Uwe‡j e¨envi Kwi Zvn‡j wØZxq †Uwe‡ji AvBwW ‡K d‡ib wK K‡i wb‡Z n‡e |

‡hgb: Avgiv BDRvi †Uwe‡ji AvBwW †K †cv÷ ‡Uwe‡j BDRvi AvBwW Kjv‡g wb‡ev Zvn‡j w¯‹gv n‡e GiKg

1. <?php
3. **use** Illuminate\Database\Migrations\Migration;
4. **use** Illuminate\Database\Schema\Blueprint;
5. **use** Illuminate\Support\Facades\Schema;
7. **class** CreatePostsTable **extends** Migration
8. {
9. /\*\*
10. \* Run the migrations.
11. \*
12. \* @return void
13. \*/
14. **public** **function** up()
15. {
16. Schema::create('posts', **function** (Blueprint $table) {
17. $table->bigIncrements('id');
18. $table->string('title');
19. $table->string('slug')->unique();
20. $table->string('sort\_des');
21. $table->longText('content');
22. $table->string('image')->nullable();
23. // $table->foreignId('user\_id')->constrained('users')->onDelete('cascade')->onUpdate('cascade');
24. // $table->foreign('user\_id')->references('id')->on('users')->onDelete('cascade')->onUpdate('cascade');
25. $table->foreignId('category\_id')->constrained('categories')->cascadeOnDelete();
26. $table->unsignedBigInteger('user\_id');
27. $table->foreign('user\_id')->references('id')->on('users')->onDelete('cascade');
28. $table->tinyInteger('publish\_status')->**default**(0);
29. $table->tinyInteger('allow\_comments')->**default**(1);
30. $table->Integer('views')->**default**(0);
31. $table->Integer('post\_order')->**default**(1);
32. $table->dateTime('published\_at');
33. $table->timestamps();
34. $table->softDeletes();
35. $table->charset = 'utf8';
36. $table->collation = 'utf8\_unicode\_ci';
37. });
38. }
40. /\*\*
41. \* Reverse the migrations.
42. \*
43. \* @return void
44. \*/
45. **public** **function** down()
46. {
47. Schema::dropIfExists('posts');
48. }
49. }
50. $table->foreignId('user\_id')->constrained('users')->onDelete('cascade')->onUpdate('cascade');
51. $table->foreign('user\_id')->references('id')->on('users')->onDelete('cascade')->onUpdate('cascade');
52. $table->foreignId('category\_id')->constrained('categories')->cascadeOnDelete();
53. $table->unsignedBigInteger('user\_id');
54. $table->foreign('user\_id')->references('id')->on('users')->onDelete('cascade');

$table->foreignId('category\_id')->constrained()->cascadeOnDelete();

GLv‡b K¨vUvMwi †Uwe‡ji AvBwW †K d‡ib wK wn‡m‡e e¨envi Kie Zvn‡j Gfv‡e w`‡jB wb‡Pi g‡Zv ‡Kv‡Wi KvR n‡q hv‡e |

$table->foreign('user\_id')->references('id')->on('users')->onDelete('cascade')->onUpdate('cascade');

**foreignId('user\_id')->constrained(),** It's the same of foreign('user\_id')->references('id')->on('users'),   
 even as   
 foreignId('user\_cid')->constrained(), it's the same of foreign('user\_cid')->references('cid')->on('users').

Schema::create('posts', function (Blueprint $table) {

$table->id();

$table->foreignId('author')->constrained('users')->onDelete('cascade')->onUpdate('cascade');

$table->timestamps();

});

Schema::create('posts', function (Blueprint $table) {

$table->increments('id');

$table->integer('author')->unsigned();

$table->timestamps();

$table->foreign('author')->references('id')->on('users')->onDelete('cascade')->onUpdate('cascade');

});

$table->id(); $table->foreignId('category\_id')->constrained('categories')->cascadeOnDelete();   
 $table->foreignId('user\_id')->constrained('users')->cascadeOnDelete()

An easy solution would be to delete the relation when the category is deleted

**Pivot migration for the relationship**

public function up()

{

Schema::create('category\_post', function (Blueprint $table) {

$table->id();

$table->unsignedInteger('post\_id');

$table->unsignedInteger('category\_id');

$table->timestamps();

$table->foreignId('category\_id')

->constrained(categories)

->onUpdate('cascade')

->onDelete('cascade');

$table->foreignId('post\_id')

->constrained(posts)

->onUpdate('cascade')

->onDelete('cascade');

});

}