Laravel for Beginners 03

Setup the MySQL project database

with an Eloquent Model Class

Configuring the DB access (MySQL)

- You should enter all needed parameters in the .env files to use the DB

```
DB_CONNECTION=mysql
DB_HOST=projects.htl-villach.at
DB_PORT=3306
DB_DATABASE=woh # use your database name
DB_USERNAME=woh # use your username
DB_PASSWORD=P@$$w0rd # use your password
```

Setup a table prefix

To distinguish between different laravel projects in the same database schema:

- Add this line in your .env file :

```
DB_TABLE_PREFIX=LAR01_ # all table names will start with LAR01_
```

- Change the marked line in the file app/config/database.php as follows:

```
'mysql' => [
    'driver' => 'mysql',
    'host' => env( key: 'DB_HOST',  default: '127.0.0.1'),
    'port' => env( key: 'DB_PORT',  default: '3306'),
    'database' => env( key: 'DB_DATABASE',  default: 'forge'),
    'username' => env( key: 'DB_USERNAME',  default: 'forge'),
    'password' => env( key: 'DB_PASSWORD',  default: ''),
    'unix_socket' => env( key: 'DB_SOCKET',  default: ''),
    'charset' => 'utf8mb4',
    'collation' => 'utf8mb4_unicode_ci',
    'prefix' => env( key: 'DB_TABLE_PREFIX',  default: ''),
    'strict' => true,
    'engine' => null,
],
```

Correct the max. string field length for MySQL

- Important: Laravel must be configured to use DB string fields with a maximum of 191 characters to use the MySQL DB at projects.htl-villach.at!
- Add the following two lines in the file `app/Providers/AppServiceProvider.php`:

```php
<?php

namespace App\Providers;

use Illuminate\Support\ServiceProvider;
use Illuminate\Support\Facades\Schema;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        Schema::defaultStringLength(191);
    }
```

Test the DB configuration

- Use Tinker in the terminal:

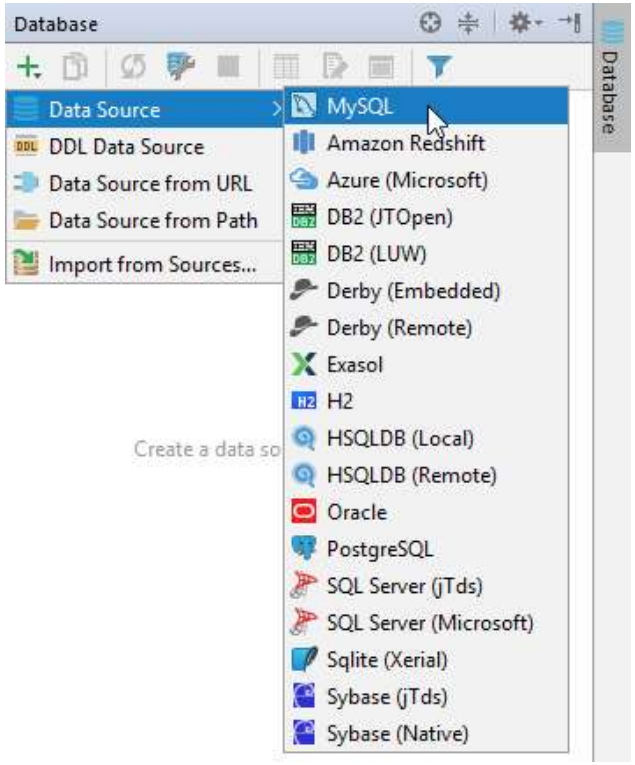
```
php artisan tinker
```

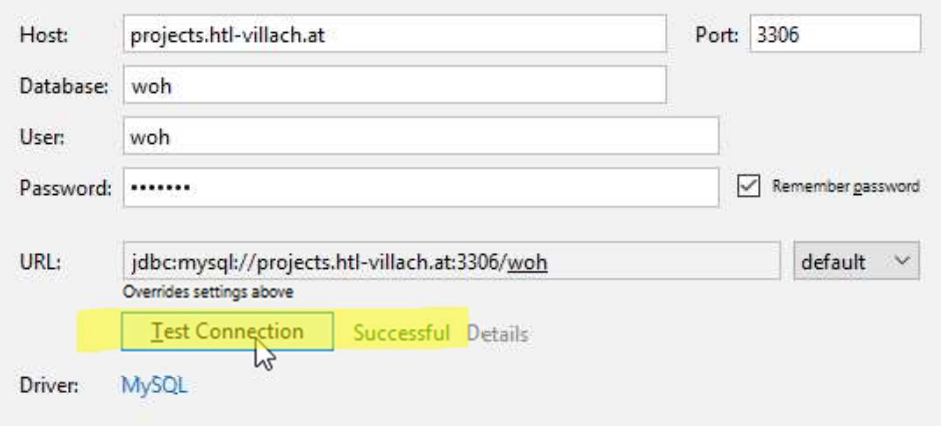
- Try to connect to the database:

```
>>> DB::connection()->reconnect();
=> null
>>> quit
```

- If you get no PDOException, the connection works!

Configure the DB Access in PHPStorm



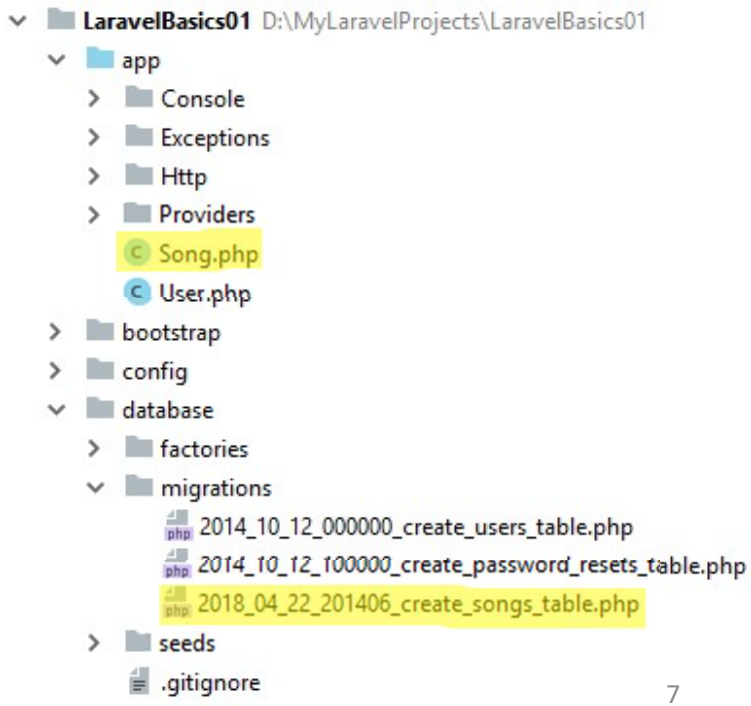


Create a model and its migration

- Create the Model class incl. migration file

```
php artisan make:model Song --migration
```

- You will get two generated files:
  - Song.php … the Model Class
    - →used to access a DB table that stores Song objects
    - →hides all SQL statements
  - Timestamp_create_songs_table.php
    - → used to create the DB table in a new/empty DB



Complete the migration file

- Complete the method up() in the generated migration file like this:

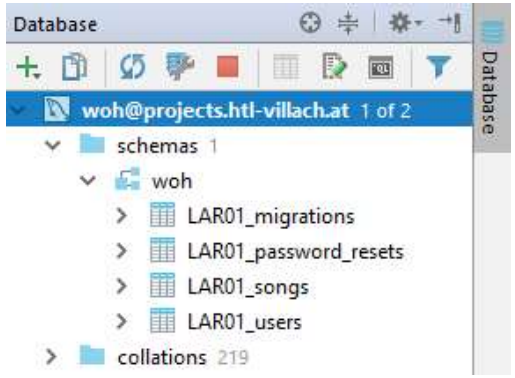
```
public function up()
{
    Schema::create('songs', function (Blueprint $table) {
        // the primary key is always the id
        $table->increments('id');

        // insert all fields of the table here
        $table->string('title', 150);
        $table->string('artist', 160);
        $table->string('album', 130);

        // the timestamps to store creation and last update
        $table->timestamps();
    });
}
```

Run the DB migration

- Migrate your database to the new DB schema:

  `php artisan migrate`

```
D:\MyLaravelProjects\LaravelBasics01>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_04_22_201406_create_songs_table
Migrated:  2018_04_22_201406_create_songs_table
```



- If you get any errors, try to do a rollback:

  `php artisan migrate:rollback`

- Or clean up your database manually by dropping all LAR01_ tables

Test the Model Class against the DB

- Use Tinker in the terminal:

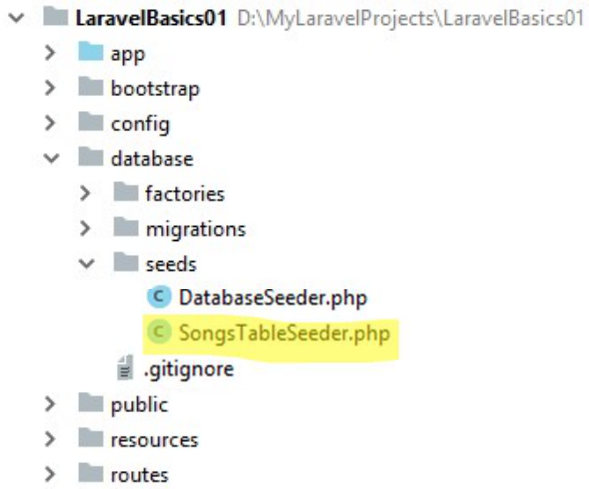
```
D:\MyLaravelProjects\LaravelBasics01>php artisan tinker
Psy Shell v0.9.3 (PHP 7.1.9 — cli) by Justin Hileman
>>> App\Song::count();
=> 0
>>> $song = new App\Song();
=> App\Song {#2311}
>>> $song->title="Magic"
=> "Magic"
>>> $song->artist="Coldplay"
=> "Coldplay"
>>> $song->album="Ghost Stories"
=> "Ghost Stories"
>>> $song->save();
=> true
>>> App\Song::count();
=> 1
>>> quit
```

	id	title	artist	album	created_at	updated_at
1	1	Magic	Coldplay	Ghost Stories	2018-04-22 ...	2018-04-22 ...
2	2	Pigs	Pink Floyd	Animals	2018-04-22 ...	2018-04-22 ...

Create a Seeder

- Laravel offers so called Seeders to fill your database with data entries you can use prepare you application to a defined state.
- Create a Seeder Class:

```
php artisan make:seeder SongsTableSeeder
```



Code and link your Seeder

You can use the Model Class „Songs“ to create and store new Songs:

- Fill the run() method with this code:
- Link your Seeder in the DataBaseSeeder.php:

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

- Run the Seeder:

```
php artisan db:seed
```

or

```
php artisan db:seed --class SongsTableSeeder
```

```php
<?php

use Illuminate\Database\Seeder;
use App\Song;

class SongsTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        // create and store some songs
        $limit = 10;
        for ($i = 1; $i <= $limit; $i++)
        {
            $song = new Song();
            $song->title = "my song title";
            $song->artist = "famous artist";
            $song->album = "some album";
            $song->save();
        }
    }
}
```

Faker

- You can use the Faker Class to generate realistic random fake data:

```
public function run()
{
    // create and store some songs
    $faker = Faker\Factory::create();
    $limit = 10;
    for ($i = 1; $i <= $limit; $i++)
    {
        $song = new Song();
        $song->title = $faker->unique()->jobTitle;
        $song->artist = $faker->firstName;
        $song->album = $faker->streetName;
        $song->save();
    }
}
```

- More details about the Faker:
  https://github.com/fzaninotto/Faker

Tips and Tricks – Code Snipplets

- Use this code to clear the content of a DB table:

```
DB::table('songs')->delete();
```

- Set the id of a song yourself:

```
for ($i = 1; $i <= $limit; $i++)
{
   $song = new Song();
   $song->id = $i;
   ...
   $song->save();
}
```

- Create the faker with german localization:

```
$faker = Faker\Factory::create('de_DE');
```