

Домашняя работа
по предмету:

Продвинутое программирование на PHP — Laravel

Выполнил: Байборodin Александр

Урок 5. Обработка запроса (Request)

Задание

Цели практической работы:

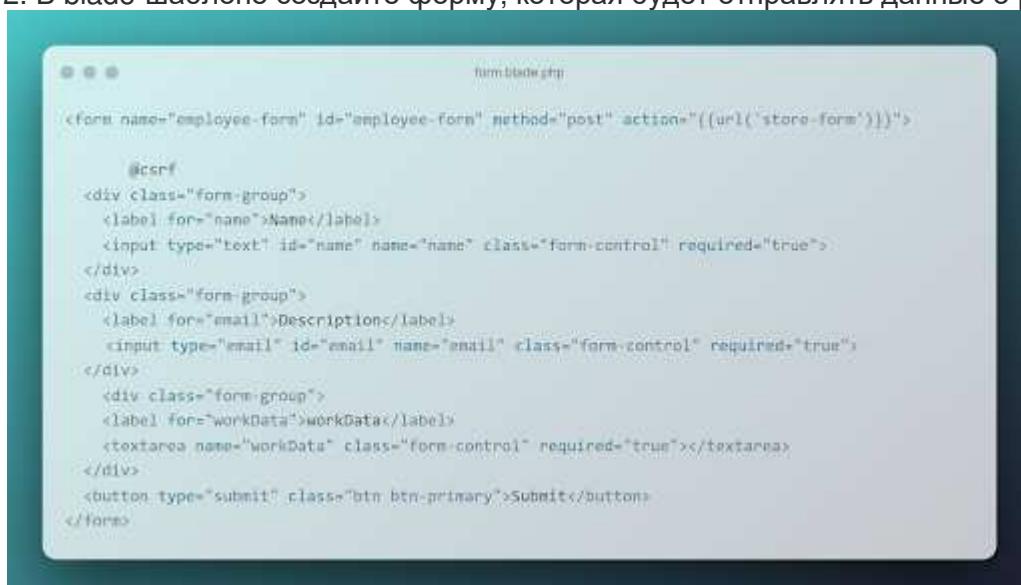
Научиться:

- использовать класс Laravel Request на практике;
- получать параметры запроса из полей ввода и адресной строки;
- передавать данные в формате JSON из полей ввода в класс Laravel Request.

Что нужно сделать:

В этой практической работе вы будете получать данные из формы и обрабатывать их в контроллере с помощью встроенных методов класса Illuminate\Http\Request.

1. В соответствующих каталогах создайте три файла:
 - blade-шаблон для создания пользовательских инпутов;
 - EmployeeController для обработки полученных данных из полей формы;
 - Route для создания динамического роутинга для отдельного работника и передачи параметра id из адресной строки.
2. В blade-шаблоне создайте форму, которая будет отправлять данные о работнике.

A screenshot of a code editor window titled 'form.blade.php'. The code is a Blade template for an HTML form. It starts with a <form> tag with attributes name='employee-form', id='employee-form', method='post', and action='{{url('store-form')}}'. Inside the form, there is a @csrf directive. The form contains three groups of controls: a text input for 'Name', an email input for 'Description', and a text area for 'workData'. Each control has a corresponding label and is wrapped in a div with class 'form-group'. The 'Name' input has class 'form-control' and required='true'. The 'Description' input has class 'form-control' and required='true'. The 'workData' text area has class 'form-control' and required='true'. At the bottom of the form is a submit button with class 'btn btn-primary' and text 'Submit'.

По аналогии с приведённым выше примером создайте ещё несколько полей ввода. Например, поля «Фамилия работника», «Занимаемая должность» и «Адрес»

проживания». Обратите внимание, что у всех полей формы есть атрибут `required="true"`. Это важно для полноты получаемых данных от клиента к серверу.

3. Создайте новый контроллер с названием `EmployeeController`. Напомним, что создавать контроллер нужно из консоли с помощью команды:



```
bash/  
php artisan make:controller EmployeeController
```

4. Внутри контроллера создайте функцию `store`, которая будет инициализировать соответствующие переменные и сохранять в них данные из вашей формы:



```
EmployeeController  
  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
class EmployeeController extends Controller  
{  
    public function index()  
    {  
        return view('get-employee-data');  
    }  
    public function store(Request $request)  
    {  
        $name = $request->input('name');  
        $email = $request->input('email');  
        //продолжите код тут...  
    }  
}
```

Добавьте все необходимые переменные в соответствии с вашими полями. Обратите внимание, что мы также создали функцию `index`, которая просто возвращает необходимый `view`.

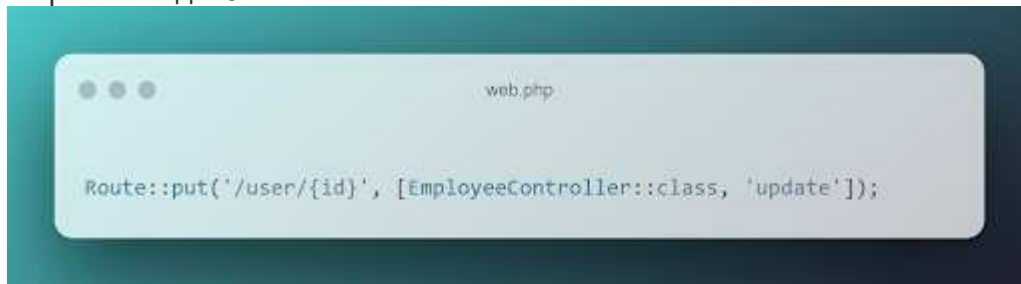
5. Как и в предыдущих занятиях, создайте необходимые роуты в файле `web.php`:



```
web.php  
  
Route::get('get-employee-data', [EmployeeController::class, 'index']);  
Route::post('store-form', [EmployeeController::class, 'store']);
```

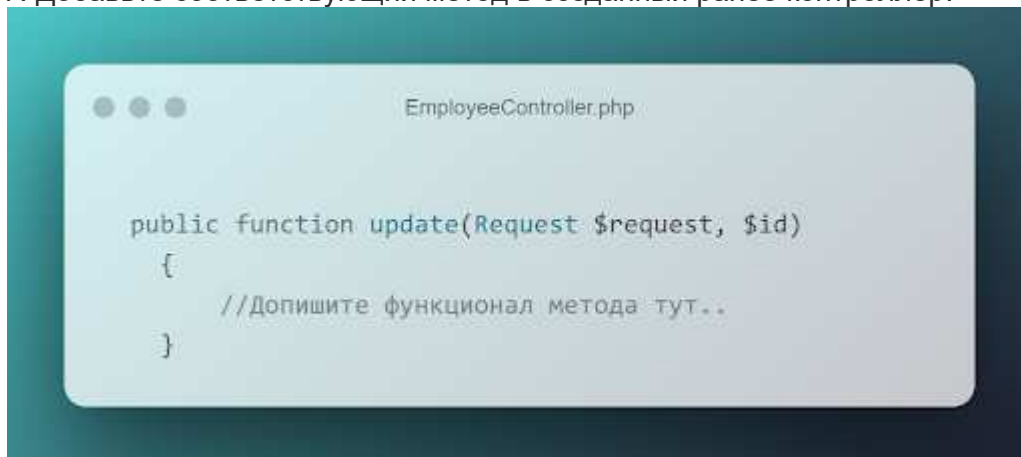
6. В файле `web.php` добавьте ещё один роут с внедрением зависимости параметров

запроса в виде id:

A screenshot of a code editor window titled 'web.php'. The code inside is a single line: `Route::put('/user/{id}', [EmployeeController::class, 'update']);`.

```
Route::put('/user/{id}', [EmployeeController::class, 'update']);
```

7. Добавьте соответствующий метод в созданный ранее контроллер:

A screenshot of a code editor window titled 'EmployeeController.php'. The code shows a public function 'update' with parameters 'Request \$request' and '\$id'. The function body is currently empty except for a comment: `//Допишите функционал метода тут..`.

```
public function update(Request $request, $id)
{
    //Допишите функционал метода тут..
}
```

Добавьте новую переменную `id`. Поместите в неё `id` из параметров запроса, обновите данные о пользователе: `name`, `email` и так далее.

8. Создайте две новые функции `getPath()`, `getUri()`, в которых необходимо получить и записать в переменную путь и URL запроса. Для этого воспользуйтесь встроенными в класс `Request` методами `$request->path()` и `$request->url()`;

Данные методы можно вызывать внутри других методов — `update` и `store`, чтобы получать служебную информацию о запросе.

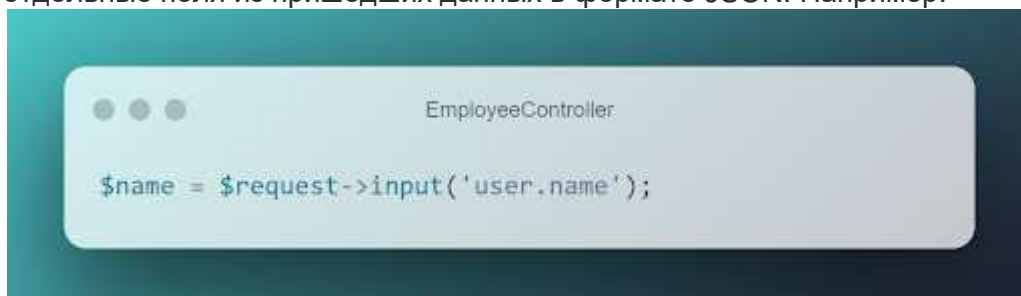
9. В форму ввода добавьте новое текстовое поле `textarea`, куда необходимо передавать данные в формате JSON, например:

A screenshot of a code editor window titled 'json'. The window contains a JSON array with one object. The object has an 'address' key, which is itself an object containing 'street', 'suite', 'city', 'zipcode', and 'geo' keys. The 'geo' key is an object with 'lat' and 'lng' keys. The code is as follows:

```
[
  {
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    }
  }
]
```

10. Обновите функции store и update. Преобразуйте полученный из запроса JSON в переменную PHP. Для этого воспользуйтесь методом `json_decode()`.

11. Создайте произвольное количество новых php переменных, в которые поместите отдельные поля из пришедших данных в формате JSON. Например:

A screenshot of a code editor window titled 'EmployeeController'. The window contains a single line of PHP code that assigns the value of the 'user.name' input field to a variable named \$name.

```
$name = $request->input('user.name');
```

Результат

Создадим проект laravel:

```
A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp
$ composer create-project laravel/laravel my-laravel-app-lesson-five
Creating a "laravel/laravel" project at "./my-laravel-app-lesson-five"
Cannot use laravel/laravel's latest version v11.6.1 as it requires php ^8.2 which is not satisfied by your platform.
Installing laravel/laravel (v10.3.3)
- Installing laravel/laravel (v10.3.3): Extracting archive
Created project in C:\laravelapp\my-laravel-app-lesson-five
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 111 installs, 0 updates, 0 removals
- Locking brick/math (0.12.1)
- Locking carbonphp/carbon-doctrine-types (2.1.0)
- Locking dflydev/dot-access-data (v3.0.3)
- Locking doctrine/inflector (2.0.10)
- Locking doctrine/lexer (3.0.1)
- Locking dragonmantank/cron-expression (v3.4.0)
- Locking egulias/email-validator (4.0.3)
- Locking fakerphp/faker (v1.24.1)
- Locking filp/whoops (2.17.0)
- Locking fruitcake/php-cors (v1.3.0)
- Locking graham-campbell/result-type (v1.1.3)
- Locking guzzlehttp/guzzle (7.9.2)
- Locking guzzlehttp/promises (2.0.4)
- Locking guzzlehttp/psr7 (2.7.0)
- Locking guzzlehttp/uri-template (v1.0.4)
- Locking hamcrest/hamcrest-php (v2.0.1)
- Locking laravel/framework (v10.48.28)
- Locking laravel/pint (v1.20.0)
- Locking laravel/prompts (v0.1.25)
- Locking laravel/sail (v1.41.0)
- Locking laravel/sanctum (v3.3.3)
- Locking laravel/serializable-closure (v1.3.7)
- Locking laravel/tinker (v2.10.1)
- Locking league/commonmark (2.6.1)
- Locking league/config (v1.2.0)
- Locking league/flysystem (3.29.1)
- Locking league/flysystem-local (3.29.0)
- Locking league/mime-type-detection (1.16.0)
- Locking mockery/mockery (1.6.12)
- Locking monolog/monolog (3.8.1)
- Locking myclabs/deep-copy (1.13.0)
```

```

- Installing theseer/tokenizer (1.2.3): Extracting archive
- Installing sebastian/lines-of-code (2.0.2): Extracting archive
- Installing sebastian/complexity (3.2.0): Extracting archive
- Installing sebastian/code-unit-reverse-lookup (3.0.0): Extracting archive
- Installing phpunit/php-code-coverage (10.1.16): Extracting archive
- Installing phar-io/version (3.2.1): Extracting archive
- Installing phar-io/manifest (2.0.4): Extracting archive
- Installing myclabs/deep-copy (1.13.0): Extracting archive
- Installing phpunit/phpunit (10.5.45): Extracting archive
- Installing spatie/error-solutions (1.1.3): Extracting archive
- Installing spatie/backtrace (1.7.1): Extracting archive
- Installing spatie/flare-client-php (1.10.1): Extracting archive
- Installing spatie/ignition (1.15.0): Extracting archive
- Installing spatie/laravel-ignition (2.9.0): Extracting archive
64 package suggestions were added by new dependencies, use 'composer suggest' to
see details.
Generating optimized autoload files
> illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

```

INFO Discovering packages.

laravel/sail	DONE
laravel/sanctum	DONE
laravel/tinker	DONE
nesbot/carbon	DONE
nunomaduro/collision	DONE
nunomaduro/termwind	DONE
spatie/laravel-ignition	DONE

82 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

INFO No publishable resources for tag [laravel-assets].

No security vulnerability advisories found.
> @php artisan key:generate --ansi

INFO Application key set successfully.

A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp

Откроем каталог с проектов в vscode:

Создадим файл resources\views\layouts\default.blade.php:

```

<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Мой сайт</title>
  <!-- Подключение Bootstrap CSS (опционально) -->
  <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>

```

```

<div class="container">
  <!-- Шапка -->
  <header class="row">
    <h1>Мой сайт</h1>
  </header>

  <!-- Основной контент -->
  <main class="row">
    @yield('content')
  </main>

  <!-- Подвал -->
  <footer class="row">
    <p>&copy; 2023 Мой сайт</p>
  </footer>
</div>

<!-- Подключение Bootstrap JS (опционально) -->
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>

```

Создадим файл resources/views/employee/form.blade.php с содержимым:

```

@extends('layouts.default')

@section('content')
  <h1>Добавить работника</h1>

  <form name="employee-form" id="employee-form" method="post" action="{{
route('employee.store') }}">
    @csrf
    <div class="form-group">
      <label for="name">Имя</label>
      <input type="text" id="name" name="name" class="form-control" required>
    </div>
    <div class="form-group">
      <label for="email">Email</label>
      <input type="email" id="email" name="email" class="form-control" required>
    </div>
    <div class="form-group">
      <label for="position">Должность</label>
      <input type="text" id="position" name="position" class="form-control" required>
    </div>
    <div class="form-group">
      <label for="workData">Дата работы</label>
      <textarea name="workData" class="form-control"></textarea>
    </div>
    <div class="form-group">
      <label for="json_data">Данные в формате JSON</label>
      <textarea name="json_data" id="json_data" class="form-control"
rows="5"></textarea>
    </div>

```

```

        <button type="submit" class="btn btn-primary">Отправить</button>
    </form>
@endsection

Создадим файл resources\views\employee\show.blade.php:
@extends('layouts.default')

@section('content')
    <h1>Информация о работнике</h1>

    @if(session('success'))
        <div class="alert alert-success">
            {{ session('success') }}
        </div>
    @endif

    <p><strong>ID:</strong> {{ $employee->id }}</p>
    <p><strong>Имя:</strong> {{ $employee->name }}</p>
    <p><strong>Email:</strong> {{ $employee->email }}</p>
    <p><strong>Должность:</strong> {{ $employee->position }}</p>
    <p><strong>Дата работы:</strong> {{ $employee->workData ?? 'Не указано' }}</p>

    <a href="{{ route('employee.index') }}" class="btn btn-primary">Добавить нового
    работника</a>
@endsection

```

Создадим контроллер с помощью Artisan:

```

MINGW64:/c:/laravelapp/my-laravel-app-lesson-five

A@LAPTOP-RHI13GV6 MINGW64 /c:/laravelapp/my-laravel-app-lesson-five
$ php artisan make:controller EmployeeController

INFO Controller [C:\laravelapp\my-laravel-app-lesson-five\app\Http\Controlle
rs\EmployeeController.php] created successfully.

A@LAPTOP-RHI13GV6 MINGW64 /c:/laravelapp/my-laravel-app-lesson-five
$

```

Создадим модель с помощью Artisan:

```

A@LAPTOP-RHI13GV6 MINGW64 /c:/laravelapp/my-laravel-app-lesson-five
$ php artisan make:model Employee -m

INFO Model [C:\laravelapp\my-laravel-app-lesson-five\app\Models\Employee.php] created suc
cessfully.

INFO Migration [C:\laravelapp\my-laravel-app-lesson-five\database\migrations\2025_02_14_1
84954_create_employees_table.php] created successfully.

A@LAPTOP-RHI13GV6 MINGW64 /c:/laravelapp/my-laravel-app-lesson-five
$ |

```

Обновим после создания модели файл миграции:

```
<?php
```



```

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up()
    {
        Schema::create('employees', function (Blueprint $table) {
            $table->id(); // id bigint(20) UN AI PK
            $table->string('name'); // name varchar(255)
            $table->string('position'); // position varchar(255)
            $table->timestamp('created_at')->nullable(); // created_at timestamp
            $table->timestamp('updated_at')->nullable(); // updated_at timestamp
            $table->string('workData')->nullable(); // workData varchar(255)
            $table->string('email')->nullable(); // email varchar(255)
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('employees');
    }
};

```

Пропишем в файле .env соединение с базой mysql:

C: > laravelapp > my-laravel-app-lesson-five >  .env

```

10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=database
15 DB_USERNAME=root
16 DB_PASSWORD=
17

```

Запустим миграции:

```
$ php artisan migrate
```

INFO Running migrations.

Проверим таблицу employees в базе database, видим она есть:

The screenshot shows a database management interface. On the left, under 'SCHEMAS', the 'employees' table is selected. Below it, the table's columns are listed: id (bigint(20) UN AI PK), name (varchar(255)), position (varchar(255)), created_at (timestamp), updated_at (timestamp), workData (varchar(255)), and email (varchar(255)). On the right, a query 'SELECT * FROM database.employees;' is executed, showing a result grid with one row of data: id 1, name John Doe, position Software Engineer, created_at 2025-02-12 18:09:56, updated_at 2025-02-12 18:09:56, workData NULL, and email example@mail.ru.

Отредактируем файл модели:

The screenshot shows a code editor with the 'Employee.php' file open. The file path is 'C:\laravelapp > my-laravel-app-lesson-five > app > Models > Employee.php'. The code defines the Employee model, extending the Model class and using the HasFactory trait. The fillable attributes are name, position, workData, and email.

```

1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Employee extends Model
9  {
10     use HasFactory;
11
12     protected $fillable = [
13         'name',
14         'position',
15         'workData',
16         'email',
17     ];
18 }
19

```

Заполним файл роутов:

<?php

use App\Http\Controllers\EmployeeController;

// Роут для отображения формы

Route::get('get-employee-data', [EmployeeController::class, 'index'])->>name('employee.index');

// Роут для обработки данных формы

Route::post('store-form', [EmployeeController::class, 'store'])->>name('employee.store');

// Роут для отображения информации о работнике

Route::get('employee/{id}', [EmployeeController::class, 'show'])->>name('employee.show');

Дополним код контроллера:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Employee;

class EmployeeController extends Controller
{
    // Метод для отображения формы
    public function index()
    {
        return view('employee.form'); // Возвращает view с формой
    }

    // Метод для обработки данных формы
    public function store(Request $request)
    {
        // Валидация данных
        $validatedData = $request->validate([
            'name' => 'required|string|max:255',
            'email' => 'required|email|max:255',
            'position' => 'required|string|max:255',
            'workData' => 'nullable|string',
        ]);

        // Сохраняем данные в базу
        $employee = Employee::create([
            'name' => $request->input('name'),
            'email' => $request->input('email'),
            'position' => $request->input('position'),
            'workData' => $request->input('workData'),
        ]);

        // Перенаправление на страницу с информацией о работнике
        return redirect()->route('employee.show', ['id' => $employee->id])->with('success',
            'Данные успешно сохранены!');
    }

    // Метод для отображения информации о работнике
    public function show($id)
    {
        // Получаем данные о работнике из базы по ID
        $employee = Employee::find($id);

        if (!$employee) {
            return redirect()->route('employee.index')->with('error', 'Работник не найден!');
        }

        return view('employee.show', compact('employee'));
    }
}
```

```

public function update(Request $request, $id)
{
    // Получаем JSON из запроса
    $jsonData = $request->input('json_data');
    $data = json_decode($jsonData, true);

    // Находим работника по ID
    $employee = Employee::find($id);

    if (!$employee) {
        return redirect()->route('employee.index')->with('error', 'Работник не найден!');
    }

    // Обновляем данные
    $employee->update([
        'name' => $data['name'] ?? $request->input('name'),
        'email' => $data['email'] ?? $request->input('email'),
        'position' => $data['position'] ?? $request->input('position'),
        'workData' => $data['workData'] ?? $request->input('workData'),
    ]);

    return redirect()->route('employee.show', ['id' => $employee->id])->with('success',
'Dанные успешно обновлены!');
}

protected function getPath(Request $request)
{
    return $request->path(); // Возвращает путь запроса (например, "store-form")
}

protected function getUrl(Request $request)
{
    return $request->url(); // Возвращает полный URL запроса (например,
"http://example.com/store-form")
}
}

```

Запустим сервер, видим, что данные работника добавляются и JSON обрабатывается:

```

A@LAPTOP-RHI13GV6 MINGW64 /c:/laravelapp/my-laravel-app-lesson-five
$ php artisan serve

```

```

[INFO] Server running on [http://127.0.0.1:8000].

```

```

Press Ctrl+C to stop the server

```

```

2025-02-14 22:45:53 ..... ~ 1s
2025-02-14 22:45:53 /favicon.ico ..... ~ 2s
2025-02-14 22:45:54 ..... ~ 4s

```

Мой сайт

Добавить работника

Имя

Иван Петров

Email

ivan@example.com

Должность

Разработчик

Дата работы

15.01.2024

Данные в формате JSON

```
{  "name": "Иван Петров",  "email": "ivan@example.com",
```

Отправить

Мой сайт

Информация о работнике

Данные успешно сохранены!

ID: 3Имя: Иван ПетровEmail: ivan@example.com

Должность: РазработчикДата работы: 15.01.2024

Добавить нового работника

MySQL Workbench

Local instance MySQL80 - W... x

FileEditViewQueryDatabaseServerToolsScriptingHelp

Navigator

SCHEMAS

Filter objects

▼ database

▼ Tables

employees

failed_jobs

migrations

password_reset_tokens

personal_access_tokens

Administration Schemas

Information

Table: employees

Columns:

id bigint(20) UN AI PK

name varchar(255)

position varchar(255)

created_at timestamp

updated_at timestamp

workData varchar(255)

email varchar(255)

tasktasktasktaskemployeesusersemployees - Tableemployees

Limit to 1000 rows

1 SELECT * FROM database.employees;

Result Grid

Filter Rows:

idnamepositioncreated_atupdated_atworkDataemail

1John DoeSoftware Engineer2025-02-12 18:09:562025-02-12 18:09:56example@mail.ru

2Иван ПетровSoftware Engineer2025-02-14 19:58:562025-02-14 19:58:562023-10-01ivan@example.com

3Иван ПетровРазработчик2025-02-14 20:06:482025-02-14 20:06:4815.01.2024ivan@example.com

NULLNULLNULLNULLNULLNULLNULL