——高性能PHP应用开发之

第11讲 Web功能支持

目录

CONTENTS

目录

CONTENTS

MySQL数据分页工作原理

■ 限定记录查询的SQL语句：select * from 表名 **limit 0, 10**;

✓ limit子句有两种形式：

- limit 10：限定获取10条记录（从序号为0的记录开始），即第1~10条记录
- limit 10, 10：限定获取10条记录（从序号为10的记录开始），即第11~20条记录

✓ 在PHP中处理分页，需要考虑几个问题：

- 每一页显示多少条记录？用变量 $pageSize 表示
- 当前是第几页？用变量 $currentPage 表示
- 当前页记录开始的下标是？ $offset = ($currentPage - 1) * $pageSize
- SQL语句：select * from 表名 limit $offset, $pageSize;

Laravel实现数据分页

在其他框架中，分页是件非常痛苦的事，Laravel的分页器集成了查询构建器和Eloquent ORM，并且开箱提供了方便的、易于使用的、基于数据库结果集的分页。分页器生成的HTML兼容Bootstrap CSS 框架。

Laravel中的查询构造器或Eloquent ORM均提供了 paginate( )方法 和simplePaginate( )方法 用来构造分页结果。这两个方法基于当前页自动设置合适的偏移（offset）和限制（limit）。默认情况下，当前页通过HTTP请求查询字符串参数?page的值判断。当然，该值由Laravel自动检测，然后自动插入分页器生成的链接中。

- **paginate**（）方法：构造常规的分页链接
- **simplePaginate**( )方法：构造简单的分页链接（不含有页码，只有“上一页”和“下一页”）

Laravel实现数据分页

- 获取分页结果

```
$results = Db::table('newss')
    ->paginate(1);
return view('news')->with('newss', $results);
```

限定每一页记录数

- 视图中展示分页信息

```
<div class="container">
    <ul>
    @foreach ($newss as $news)
        <li>{{ $news->title }}</li>
    @endforeach
    </ul>
</div>
{{ $newss->links() }}
```

分页码

- 浏览器中查看分页效果

news_title

« 1 2 »

目录

CONTENTS

PHP中Session的工作原理

■ Session

- ✓ 本质：session默认情况下是存储在服务器端的一个文件，对session的读写，其实就是对文件的读写，只不过PHP中把session的读写操作进行了封装。
- ✓ 作用：在不同的HTTP请求间共享session数据
- ✓ 服务器端标识session：针对不同客户端，服务器通过 session_id 来标识不同客户端所对应的session文件；显然 session_id 需要客户端发送给服务器以识别，故session_id一般保存在客户端cookie中。
- ✓ session的使用流程：

① 开启session　② 设置session数据　③ 判断session是否有效　④ 销毁session

Laravel中Session相关配置

- Laravel中session的核心配置文件为 **/config/session.php** 文件

✓ 配置session的存储方式：默认情况下，Laravel使用的session驱动为文件驱动，在生产环境中，可能会考虑使用memcached或者redis驱动以便获取更快的session性能。

Laravel中支持的session存储方式主要有：

- file – session数据存储在 storage/framework/sessions目录下；
- cookie – session数据存储在经过加密的安全的cookie中；
- database – session数据存储在数据库中
- memcached / redis – session数据存储在memcached/redis中；
- array – session数据存储在简单PHP数组中，在多个请求之间是非持久化的

```
'driver' => env('SESSION_DRIVER', 'file'),
```

Laravel中Session相关配置

- Laravel中session的核心配置文件为 **/config/session.php** 文件

✓ 配置session的过期时间：

```
'lifetime' => 120,
```

120秒后，session自动过期

```
'expire_on_close' => false,
```

浏览器关闭时，session是否过期

✓ 数据库作为session驱动时，设置数据表名称：

```
'table' => 'sessions',
```

✓ session_name的名称：

```
'cookie' => 'laravel_session',
```

Laravel中使用Session

- Laravel中使用session，不需要手动开启session；session是在程序启动时自动开启的。
- 存储Session数据

```
//通过put方法
$request->session()->put('key', 'value');

//通过全局辅助函数
session(['key' => 'value']);
```

- 获取Session数据

```
// 从session中获取数据...
$value = session('key');
// 指定默认值...
$value = session('key', 'default');
```

- 判断Session数据的有效性

```
if ($request->session()->has('users')) {
    //
}
```

- 销毁Session数据

```
$request->session()->forget('key');
$request->session()->flush();
```

注意：使用session时，不要使用 dd( )函数或exit;语句；否则会导致session保存不成功。

使用数据库存储Session

- 使用数据库存储Session，主要是添加 session 数据表，其它基本操作同文件存储方式。
- 构建session表迁移文件：**php artisan session:table**
  - ✓ 该命令将自动在 /database/migrations/目录下创建好session表的迁移文件
- 执行数据库迁移：php artisan migrate

```php
class CreateSessionsTable extends Migration {
    /**
    public function up() {
        Schema::create('sessions', function (Blueprint $table) {
            $table->string('id')->unique();
            $table->integer('user_id')->nullable();
            $table->string('ip_address', 45)->nullable();
            $table->text('user_agent')->nullable();
            $table->text('payload');
            $table->integer('last_activity');
        });
    }

    /**
    public function down()
    {
        Schema::dropIfExists('sessions');
    }
}
```

目录

CONTENTS

验证码

Laravel没有提供验证码功能，可以使用扩展库的形式引入验证码。

- gregwar/captcha库：
  - ✓ 下载扩展库：composer require gregwar/captcha
  - ✓ 使用扩展库：使用Composer类库方式使用即可
  - ✓ 官网：https://packagist.org/packages/gregwar/captcha

```php
<?php

use Gregwar\Captcha\CaptchaBuilder;

$builder = new CaptchaBuilder;
$builder->build();
```

```php
<?php

// Example: storing the phrase in the session to test for the user
// input later
$_SESSION['phrase'] = $builder->getPhrase();
```

```php
if($builder->testPhrase($userInput)) {
    // instructions if user phrase is good
}
else {
    // user phrase is wrong
}
```

验证码

- mews/captcha库：
- ✓ 下载扩展库：composer require mews/captcha
- ✓ 使用扩展库：
  - 配置 config/app.php 文件
  - 生成配置文件：php artisan vendor:publish --tag=***
  - 使用验证码
- ✓ 官网：https://packagist.org/packages/mews/captcha

```
$form .= '<p>' . captcha_img() . '</p>';
```

```
$validator = Validator::make(Input::all(), $rules);
if ($validator->fails())
{
    echo '<p style="color: #ff0000;">Incorrect!</p>';
}
else
{
    echo '<p style="color: #00ff30;">Matched :)</p>';
}
```

目录

CONTENTS

文件上传准备

■ HTML准备

- ✓ 表单提交方式：POST表单
- ✓ <form>标签 enctype属性：multipart/form-data
- ✓ 表单控件：<input type="file" name="***" />

■ PHP基础

- ✓ 配置项：upload_max_filesize、max_file_uploads、post_max_size
- ✓ 获取上传的文件：$_FILES[ ]
- ✓ 保存上传的文件：move_uploaded_file( )

文件上传的处理

■ 获取上传的文件

- ✓ 请求对象的hasFile( )方法：判断HTTP请求中是否有文件上传信息
- ✓ 请求对象的file( )方法：获取上传的文件，返回 Illuminate\Http\UploadedFile 对象
  - 该类继承 SplFileInfo 类，可以使用 SplFileInfo 类的所有公有方法
  - UploadedFile::isValid( )方法：判断文件上传是否成功
  - UploadedFile::getClientOriginalName( )：获取原始文件名

■ 保存上传的文件

- ✓ UploadedFile::store( )方法：接收一个文件保存的相对路径（相对于文件系统配置的根目录），该路径不应该包含文件名，因为文件名会通过对文件内容进行md5自动生成
- ✓ UploadedFile::storeAs( )方法：该方法接收保存路径、文件名和磁盘名作为参数

文件上传配置项

■ 配置文件：config/filesystems.php

- ✓ 默认加载的文件驱动：'default' => 'local',
- ✓ 驱动引擎配置：

```
'local' => [
    'driver' => 'local',
    // 文件上传根目录
    'root' => storage_path('app'),
],
```

多文件上传

■ 数组形式

- ✓ HTML表单： <input type="file" name="photo[]" />
- ✓ Laravel中获取： $files = $request->file('photo');
- ✓ 处理每一个文件：foreach循环遍历

```php
foreach ($files as $file) {
    $file->storeAs('images', $file->getClientOriginalName());
}
```

■ 不同name属性形式

- ✓ HTML表单： <input type="file" name="photo" />
  <input type="file" name="photo2" />
- ✓ Laravel中获取： $files = $request->file();
- ✓ 处理每一个文件：foreach循环遍历

```
array:2 [▼
  "photo" => UploadedFile {#184 ▶}
  "photo2" => UploadedFile {#187 ▶}
]
```

感谢聆听！

THANK YOU FOR YOUR ATTENTION