DOI 10.52209/1609-1825_2022_1_253 УДК 004.4'233

Анализ структуры сторонних данных на языке PHP

[1]**КУВАРДИН Максим Вячеславович,** *магистрант, maxim@kuvard.in,*
[1]**ЯВОРСКИЙ Владимир Викторович,** *д.т.н., профессор, yavorskiy-v-v@mail.ru,*
[1]***КОККОЗ Махаббат Мейрамкызы,** *к.п.н., зав. кафедрой, makhabbat_k@bk.ru,*
[2]**КАЦАГА Татьяна Яковлевна,** *ведущий инженер, tkatsaga@itasca.ca,*
[1]*Карагандинский технический университет, Казахстан, 100027, Караганда, пр. Н. Назарбаева, 56,*
[2]*Консалтинговая компания Itasca, Канада, Торонто,*
**автор-корреспондент.*

Аннотация. *Разработчики серверных программных решений в процессе своей деятельности могут сталкиваться с необходимостью использования данных из сторонних источников. Такими источниками могут быть, например, интерфейсы разработки приложений (API) либо веб-страницы со структурированными данными. Некоторые источники данных при этом сопровождаются технической документацией. Однако документация не всегда обладает всей полнотой необходимой информации. В статье рассматривается разработка прикладной библиотеки на языке PHP. Описаны существующие типы данных языка, а также введены частные типы. Рассматриваются установка и примеры использования данной разработки.*

Ключевые слова: *PHP, структура данных, типы данных, API, анализ данных, мониторинг, надежность ПО, ИТ-система, библиотека, инфраструктура, сервис, сервер, информация, безопасность, качество, структура, эффективность.*

Разработчики серверных программных решений в процессе своей деятельности могут сталкиваться с необходимостью использования данных из сторонних источников. Такими источниками могут быть, например, интерфейсы разработки приложений (API) либо веб-страницы со структурированными данными. Некоторые источники данных при этом сопровождаются технической документацией. Однако документация не всегда обладает всей полнотой необходимой информации.

Для предотвращения возможных ошибок разработчик должен опираться на следующую информацию об используемых сторонних данных:

1. Наименования возможных полей.
2. Типы данных каждого поля.
3. Возможность иметь пустое значение (nullable-признак поля).
4. Обязательность наличия каждого поля.
5. Диапазон возможных значений – для числовых типов.
6. Диапазон возможной длины строки – для строкового типа.
7. Диапазон возможного количества элементов – для массивов.
8. Ассоциативность/диссоциативность – для массивов.
9. Примеры возможных значений для каждого поля [1].

В данной работе описывается разработка прикладной библиотеки для анализа типов и структуры сторонних данных на языке программирования PHP.

В языке программирования PHP существуют следующие типы данных, объединённые в группы.

Скалярные типы: логический (bool), целочисленный (int), числа с плавающей точкой (float или double) и строковый (string) [2]. Представляют собой элементарные типы данных. Присутствуют в большинстве языков, однако, в отличие от реализации в некоторых языках (например JS), данные этих типов не являются экземплярами соответствующих классов. Для работы с данными скалярных типов в языке PHP реализованы специальные операторы и функции.

Смешанные типы: массив (array), объект (object), callback-функции (callable) и итерируемый тип (iterable) [3]. Массив и объект представляют собой структурированные типы для упрощения разработки и для логического объединения данных (также объекты являются частью реализации парадигмы объектно-ориентированного программирования). Callback-функции (или функции обратного вызова) – обычные, анонимные и стрелочные функции с поддержкой обратного вызова. В отличие от остальных типов данных, крайне редко используются для обмена данными между информационными системами. Iterable – псевдоним для данных, которые мож-

но перебирать с помощью php-цикла foreach, а также использовать в генераторах с оператором yield from. Такими данными являются массивы и объекты классов, реализующих интерфейс Traversable. Типы данных callable и iterable не будут рассматриваться в данной статье из-за специфичности их применения.

Наконец, специальные типы – переменная без значения (null) и ссылка на внешний ресурс (resource) [4]. В отличие от других типов, null редко бывает единственным возможным значением переменной или поля. Чаще всего null используется в паре с другими типами. В таком случае переменная или поле приобретает свойство nullable. Важно точно устанавливать nullable переменные и прямо указывать это в коде во избежание непредвиденных ошибок при исполнении. Переменные и поля типа resource в свою очередь хранят ссылки на внешние ресурсы, которые создаются и используются специальными функциями. В состав ссылки на ресурс входят тип ресурса (строка) и его идентификатор (целое число).

Помимо существующих типов данных также важно отслеживать условные типы: пустая строка, пустой массив, числовая строка.

Для анализа сторонних данных была разработана и выгружена в открытый доступ библиотека FieldsScanner (с англ. «сканер полей»). Она предоставляет разработчикам широкие возможности для изучения структуры сторонних данных. Библиотека расположена в GitHub-репозитории и доступна всем желающим по адресу https://github.com/kuvardin/FieldsScanner.

Библиотека устанавливается как сторонний пакет посредством менеджера пакетов Composer. Для установки достаточно в папке с проектом выполнить команду «composer require kuvardin/fields-scanner». Если на компьютер установлен менеджер пакетов Composer и команда composer корректно обрабатывается командной оболочкой, пакет будет успешно установлен в папку vendor.

Для описания каждого типа данных были созданы соответствующие классы в пространстве имен Kuvardin\FieldsScanner\Types, расширяющие абстрактный класс Kuvardin\FieldsScannes\Type. В каждом из этих классов реализованы следующие методы, которые требует родительский класс Type:

1. public static function checkValue($value): bool – проверка принадлежности значения данному типу;

2. public static function getCode(): string – получение кода текущего типа (Integer, Sting, Bool...);

3. public function addValue($value): void – обработка еще одного возможного значения для поля данного типа.

Каждое поле данных (переменная, элемент массива, поле объекта) представлено в виде экземпляра класса Kuvardin\FieldsScannes\Field, содержащего следующие свойства:

1. array $types – возможные типы данных значения поля;

2. int $null_number – количество случаев, когда поле имело пустое значение (null);

3. int $not_exists_number – количество случаев, когда элемент массива или поле класса не существовало;

4. int $exists_number = 0 количество случаев, когда поле существовало.

Исходя из значений перечисленных свойств можно получить общие сведения о поле.

Для сканирования данных необходимо создать экземпляр класса Kuvardin\FieldsScanner\FieldsScanner, а затем передать в него данные, вызвав метод scan(). При этом важно передать как можно больше вариантов однородных данных с различным содержимым. Например, если данные берутся из метода стороннего интерфейса разработки приложений (API), который принимает на входе номер страницы, желательно перебрать несколько страниц.

После успешного сканирования, поле result созданного объекта станет экземпляром класса Field. Именно в нем будет сохраняться результат работы сканера после каждой итерации сканирования.

Рассмотрим пример использования данного решения. В качестве данных возьмем открытый API проекта Google Books. Данный проект располагает большим объемом информации о книгах и других печатных изданиях.

Для поиска по базе сторонним разработчикам необходимо выполнить HTTP-запрос по следующему адресу: https://www.googleapis.com/books/v1/volumes?q=текст_запроса [5].

Получим результаты для различных текстов запроса и просканируем через FieldsScanner (рисунок 1).

В результате выполнения данного скрипта пользователю отобразится информация в доступном виде (рисунок 2).

В результате отобразится результат работы в форматированном виде, удобном для человеческого восприятия. Опираясь на полученные данные, сторонний программист сможет лучше ознакомиться с данными, принимаемыми из внешних серверов [6].

Рассмотрим другой пример: анализ данных, возвращаемых API сервиса получения информации об IP-адресах. Далее приведен исходный код примера (рисунок 3) и результат его выполнения (рисунок 4).

На примерах виден общий алгоритм работы с библиотекой. Для начала пользования сканера создается экземпляр класса FieldsScanner. Далее производится запрос к сторонним API, и дальнейшее декодирование полученных данных, при его необходимости. Полученные данные передаются методу scan() экземпляра класса FieldsScanner.

Обратите внимание, что получение однородных данных и их передачу в сканер необходимо производить несколько раз. Это повышает точ-

```php
<?php

require 'vendor/autoload.php';

$scanner = new Kuvardin\FieldsScanner\FieldsScanner;
$queries = ['Саган', 'Фейнман', 'Докинз', 'Невзоров'];

foreach ($queries as $query) {
    echo "Searching: $query\n";
    $url = 'https://www.googleapis.com/books/v1/volumes?' .
        http_build_query(['q' => $query]);
    $response = file_get_contents($url);
    if ($response === false) {
        continue;
    }

    $response_json = json_decode($response);
    $scanner->scan($response_json);
}

echo $scanner->result->getInfo();
```

Рисунок 1 – Пример использования библиотеки

```
Types: assoc_array (4)
Type assoc_array:
   [kind] - string (4)
      Type string:
         Length: 13-13 (cannot be empty)
   [totalItems] - int (4)
      Type int:
         Values: 219 - 485
   [items] - dissoc_array (4)
      Type dissoc_array:
         Length: 10-10 (cannot be empty)
         Child - assoc_array (40)
            Type assoc_array:
               [kind] - string (40)
                  Type string:
                     Length: 12-12 (cannot be empty)
               [id] - string (40)
                  Type string:
                     Length: 12-12 (cannot be empty)
```

Рисунок 2 – Результат выполнения скрипта

ность их описания.

В качестве входных данных подойдут не только ответы сторонних API, закодированные в JSON или XML форматах, но и любые другие виды форматированных данных [7].

Представление результата в виде объектов классов дает широкий спектр возможностей, включая автоматическую генерацию классов-оберток и документации.

В результате разработки библиотеки kuvardin/FieldsScanner удалось создать удобный и необходимый современным разработчикам инструмент для получения подробной информации о структуре и типах сторонних данных. Были изучены и применены тонкости разработки на языке программирования PHP.

Разработанная библиотека имеет потенциал дальнейшего развития. Размещение ее в отдельном репозитории на GitHub дает возможность сторонним разработчикам принимать участие в разработке проекта.

```php
<?php

require 'vendor/autoload.php';
$scanner = new Kuvardin\FieldsScanner\FieldsScanner;

$limit = (int)($argv[1] ?? 10);
for ($i = 0; $i < $limit; $i++) {
    $ip = random_int(1, 254) . '.' .
          random_int(1, 254) . '.' .
          random_int(1, 254) . '.' .
          random_int(1, 254);
    echo $ip, PHP_EOL;

    $url = "http://ip-api.com/json/$ip";

    $response = file_get_contents($url);
    if ($response === false) continue;

    $response_json = json_decode($response);
    $scanner->scan($response_json);
    sleep(1);
}

echo $scanner->result->getInfo();
```

Рисунок 3 – Анализ результата работы стороннего API

```
Types: assoc_array (20)
Type assoc_array:
    [status] string (20)
        Type string:
            Length: 4-7 (cannot be empty)
            Examples: success (16), fail (4)
    [country] string (16), not_exist (4)
        Type string:
            Length: 5-14 (cannot be empty)
            Examples: China (3), United States (8)
    [countryCode] string (16), not_exist (4)
        Type string:
            Length: 2-2 (cannot be empty)
            Examples: CN (3), US (8), GB (1)
    [region] string (16), not_exist (4)
        Type string:
            Length: 2-3 (cannot be empty)
            Examples: ZJ (1), NY (2), CQ (1)
    [regionName] string (16), not_exist (4)
        Type string:
            Length: 4-14 (cannot be empty)
            Examples: Zhejiang (1), New York (2)
    [city] string (16), not_exist (4)
        Type string:
            Length: 4-17 (cannot be empty)
            Examples: Hangzhou (1), New York (2)
    [zip] string (16), not_exist (4)
        Type string:
            Length: 3-8 (may be empty)
            Examples: 10118 (1), W1B (1)
    [lat] float (16), not_exist (4)
        Type float:
            Values: 29.5514 - 52.2839
            Examples: 30.2674 (1), 40.7126 (1)
```

Рисунок 4 – Результат работы скрипта

СПИСОК ЛИТЕРАТУРЫ

1. Мэтт Зандстра. PHP: объекты, шаблоны и методики программирования = PHP Objects, Patterns and Practice, Third Edition. – 3-е издание. – М.: Вильямс, 2010. – 560 с. – ISBN 978-5-8459-1689-1.
2. Кристиан Дари, Эмилиан Баланеску. PHP и MySQL: создание интернет-магазина = Beginning PHP and MySQL E-Commerce: From Novice to Professional. – М.: Вильямс, 2010. – ISBN 978-5-8459-1602-0.
3. Джейсон Ленгсторф. PHP и jQuery для профессионалов = Pro PHP and jQuery. – М.: Вильямс, 2010. – 352 с. – ISBN 978-5-8459-1693-8.
4. Квентин Зервас. Web 2.0: создание приложений на PHP = Practical Web 2.0 Applications with PHP. – М.: Вильямс, 2009. – 544 с. – ISBN 978-5-8459-1590-0.
5. Кузнецов Максим, Симдянов Игорь. PHP 5/6. – СПб: «БХВ-Петербург», 2009. – 1024 с. – ISBN 978-5-9775-0304-4.
6. Эд Леки-Томпсон, Алек Коув, Стивен Новицки, Хьяо Айде-Гудман. PHP 5 для профессионалов = Professional PHP 5. – М.: Диалектика, 2006. – 608 с. – ISBN 0-7645-7282-2.

PHP-дегі бөгде мәліметтер құрылымын талдау

[1]КУВАРДИН Максим Вячеславович, *магистрант, maxim@kuvard.in,*
[1]ЯВОРСКИЙ Владимир Викторович, *т.ғ.д., профессор, yavorskiy-v-v@mail.ru,*
[1]*КОККОЗ Махаббат Мейрамкызы, *п.ғ.к., кафедра меңгерушісі, makhabbat_k@bk.ru,*
[2]КАЦАГА Татьяна Яковлевна, *жетекші инженер, tkatsaga@itasca.ca,*
[1]Қарағанды техникалық университеті, Қазақстан, 100027, Қарағанды, Н. Назарбаев даңғылы, 56,
[2]Itasca консалтингтік компаниясы, Канада, Торонто,
**автор-корреспондент.*

Аңдатпа. *Серверлік бағдарламалық жасақтама жасаушылар өз қызметі барысында бөгде көздерден алынған деректерді пайдалану қажеттілігіне тап болуы мүмкін. Мұндай көздер, мысалы, бағдарламалық жасақтама интерфейстері (API) немесе құрылымдық деректері бар веб-парақтар болуы мүмкін. Кейбір деректер көздеріне техникалық құжаттама қоса беріледі. Алайда, құжаттамада барлық қажетті мәліметтер бола бермейді. Мақалада PHP тілінде қолданбалы кітапхананың дамуы туралы айтылады. Берілген тілдің қолданыстағы деректер типтері, оған қоса, жеке типтер сипаттамасы енгізілген. Әзірленген бағдарламаны орнату және қолдану мысалдары қарастырылған.*

Кілт сөздер: *PHP, мәліметтер құрылымы, мәліметтер типтері, API, деректерді талдау, мониторинг, бағдарламалық қамтамасыз етудің сенімділігі, АТ жүйесі, кітапхана, инфрақұрылым, сервис, сервер, ақпарат, қауіпсіздік, сапа, құрылым, тиімділік.*

Structure Analysis of External Data via PHP Programming Language

[1]KUVARDIN Maxim, *master student, maxim@kuvard.in,*
[1]YAVORSKIY Vladimir, *Dr. of Tech. Sci., Professor, yavorskiy-v-v@mail.ru,*
[1]*KOKKOZ Makhabbat, *Cand. of Ped. Sci., Head of Department, makhabbat_k@bk.ru,*
[2]KATSAGA Tatyana, *Senior Engineer, tkatsaga@itasca.ca,*
[1]Karaganda Technical University, Kazakhstan, 100027, Karaganda, N. Nazarbayev Avenue, 56,
[2]Itasca Consulting Group, Canada, Toronto,
**corresponding author.*

Abstract. *Developers of server application can meet a need in using the data from external sources. For example: API or web pages with structured information. Moreover, some sources might be supported by documentation. However, the documentation is not always full and helpful. The article discusses the development of an application library in PHP. The existing data types of the language are described, and private types are introduced. The installation and examples of using this development are considered.*

Keywords: *PHP, data structure, data types, API, data analysis, monitoring, reliability of the software, IT system, library, infrastructure, service, server, data, security, quality, structure, efficiency.*

REFERENCES

1. Mett Zandstra. PHP: ob"ekty, shablony i metodiki programmirovaniya = PHP Objects, Patterns and Practice, Third Edition. – 3-e izdanie. – Moscow: Vil'yams, 2010. – 560 p. – ISBN 978-5-8459-1689-1.
2. Kristian Dari, Emilian Balanesku. PHP i MySQL: sozdanie internet-magazina = Beginning PHP and MySQL E-Commerce: From Novice to Professional. – Moscow: Vil'yams, 2010. – ISBN 978-5-8459-1602-0.
3. Dzhejson Lengstorf. PHP i jQuery dlya professionalov = Pro PHP and jQuery. – Moscow: Vil'yams, 2010. – 352 p. – ISBN 978-5-8459-1693-8.
4. Kventin Zervas. Web 2.0: sozdanie prilozhenij na PHP = Practical Web 2.0 Applications with PHP. – Moscow: Vil'yams, 2009. – 544 p. – ISBN 978-5-8459-1590-0.
5. Kuznecov Maksim, Simdyanov Igor'. PHP 5/6. – Saint Petersburg: «BHV-Peterburg», 2009. – 1024 p. – ISBN 978-5-9775-0304-4.
6. Ed Leki-Tompson, Alek Kouv, Stiven Novicki, H'yao Ajde-Gudman. PHP 5 dlya professionalov = Professional PHP 5. – Moscow: Dialektika, 2006. – 608 p. – ISBN 0-7645-7282-2.