УДК 004

Использование REST архитектуры для построения взаимодействия компонентов в веб-сервисе

Круглик Роман Игоревич
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
В статье рассматривается построение REST архитектуры для взаимодействия компонентов системы. Рассмотрены 5 основных запросов к данным, которые используются при работе с API.
Ключевые слова: REST, архитектура, API.

Using REST architecture to build the interaction of components in a web service

Kruglik Roman Igorevich
Sholom-Aleichem Priamursky State University
Student

Abstract
In article discusses the construction of a REST architecture for the interaction of system components. Five basic data requests that are used when working with the API are considered.
Keywords: REST, architecture, API.

Каждый день создаются новые веб-сервисы, которые необходимо правильно проектировать. При проектировании веб-сервиса, нужно учитывать различные факторы. Одним из них является построение взаимодействующих компонентов системы. Для этого существует специальная архитектура REST, которая помогает построить взаимодействие данных в приложении.

REST - архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (интернет-магазины, поисковые системы, прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине.

В сети Интернет вызов удалённой процедуры может представлять собой обычный HTTP-запрос (обычно «GET» или «POST»; такой запрос

называют «REST-запрос»), а необходимые данные передаются в качестве параметров запроса.

Для веб-служб, построенных с учётом REST (то есть не нарушающих накладываемых им ограничений), применяют термин «RESTful».

В отличие от веб-сервисов (веб-служб) на основе SOAP, не существует «официального» стандарта для RESTful API. Дело в том, что REST является архитектурным стилем, в то время как SOAP является протоколом. Несмотря на то, что REST не является стандартом сам по себе, большинство RESTful-реализаций используют такие стандарты, как HTTP, URL, JSON и XML.

Всего существует пять типов запросов:

1. GET - используется для поиска информации со стороны сервера.
2. POST - вносит новую информацию на сервер.
3. PUT и PATCH – используются для обновления определенной информации на сервере.
4. DELETE – удаляет указанную сущность из базы или сигнализирует об ошибке.

Исследования в области использование архитектуры REST для проектирования веб-сервисов актуальные и по сей день. В статье К.В. Федотова, А. Хоук [1] приводится обзор вариантов использования REST в современной архитектуре web-приложений. В.А. Егунов, А.А. Князев [2] расписывают проектирование оптимизированной архитектуры web-клиента, взаимодействующего с REST API. В статье Ю.А. Зотова, И.Д. Котилевец [3] расписана разработка архитектуры REST API для взаимодействия с сервисами приложения. С.А. Дроздов, В.Е. Луканина [4] рассказывают об особенностях проектирования серверного и клиентского программного обеспечения web-сайта с использованием REST-архитектуры.

В данной статье рассматривается создание шаблона для построения REST архитектуры.

Представим, что мы создаём интернет-магазин по продаже фруктов. В данном магазине имеются 2 сущности – фрукты и пользователи. Рассмотрим построение пяти вышеупомянутых запросов.

Для начала настроим наш серверный файл .htaccess так, чтобы все запросы перенаправлялись на файл index.php. Именно он и будет заниматься извлечением данных (см. рис. 1).

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.+)$ index.php?q=$1 [L,QSA]
```

Рисунок 1. Файл .htaccess

Теперь перейдём на главный файл index.php. В нём должен содержаться роутинг приложения (см. рис. 2).

```php
<?php
$method = $_SERVER['REQUEST_METHOD'];
$formData = getFormData($method);
function getFormData($method) {
    if ($method === 'GET') return $_GET;
    if ($method === 'POST') return $_POST;
    $data = array();
    $exploded = explode( delimiter: '&', file_get_contents( filename: 'php://input'));
    foreach($exploded as $pair) {
        $item = explode( delimiter: '=', $pair);
        if (count($item) == 2) {
            $data[urldecode($item[0])] = urldecode($item[1]);
        }
    }
    return $data;
}
$url = (isset($_GET['q'])) ? $_GET['q'] : '';
$url = rtrim($url, charlist: '/');
$urls = explode( delimiter: '/', $url);
$router = $urls[0];
$urlData = array_slice($urls, offset: 1);
include_once 'routers/' . $router . '.php';
route($method, $urlData, $formData);
?>
```

Рисунок 2. Файл index.php

Сначала считывает метод, по которому передаётся запрос, далее исходя из того какой метод выполняется действие. Если это POST или GET, то ничего не происходит. Если же это PUT, PATCH или DELETE, то тут необходимо правильно достать данные из массива с которыми мы будем работать. После чего считывается нужные параметры, подключается файл обработчик и вызывается функция route. Так как мы работаем с фруктами, файл обработчик будет называться – fruits.php. (см. рис. 3).

```php
<?php
function route($method, $urlData, $formData) {
    if ($method === 'GET' && count($urlData) === 1) {
        $fruitId = $urlData[0];
        echo json_encode(array(
            'method' => 'GET',
            'id' => $fruitId,
            'fruit' => 'Лимон',
            'price' => '7 рублей'
        ));
        return;
    }

    header( string: 'HTTP/1.0 400 Bad Request');
    echo json_encode(array(
        'error' => 'Так не пойдёт. Ошибка 404'
    ));
}
?>
```

Рисунок 3. Файл fruits.php

Если вызван метод GET, то функция вернет массив, в котором будет указан id из ссылки и другие атрибуты. Смысл в том, что через GET запрос мы можем обратиться к базе данных и достать оттуда любой нужный нам элемент. Чаще всего такой метод используется веб-разработчиками и называется API – это набор правил, при помощи которых одно приложение может взаимодействовать с другим. Так же данный метод необходим если сервис имеет мобильное приложение, в котором необходимо показывать ту же информацию, что и на веб-сервисе. Проверим работоспособность такого метода извлечения информации через curl. Впишем в консоль «curl -X GET http://test/fruit/15 -i» (см. рис. 4).

```
E:\OSPanel\domains\test>curl -X GET http://test/fruits/10 -i
HTTP/1.1 200 OK
Date: Thu, 29 Aug 2019 11:08:11 GMT
Server: Apache
Content-Length: 62
Content-Type: text/html; charset=UTF-8

{"method":"GET","id":"10","fruit":"limon","price":"7 rub"}
```

Рисунок 4. Результат запроса GET

Все ответы приходят в формате JSON и вот мы видим результат. Теперь запрограммируем обработку метода POST на создание новой сущности. (см. рис. 5).

```
if ($method === 'POST' && empty($urlData)) {
    echo json_encode(array(
        'method' => 'POST',
        'id' => rand(1, 100),
        'formData' => $formData
    ));
    return;
}
```

Рисунок 5. Обработчик запроса POST

Теперь при запросе добавляется новый продукт. Впишем в консоль «curl -X POST http://test/fruits/ --data "fruit=pear&price=40" -i» (см. рис. 6).

```
E:\OSPanel\domains\test>curl -X POST http://test/fruits/ --data "fruit=pear&price=40" -i
HTTP/1.1 200 OK
Date: Thu, 29 Aug 2019 11:25:57 GMT
Server: Apache
Content-Length: 70
Content-Type: text/html; charset=UTF-8

{"method":"POST","id":32,"formData":{"fruit":"pear","price":"40"}}
```

Рисунок 6. Результат выполнения запроса POST

В результате видим добавление груши за 40 рублей. Так же при помощи запросов PUT или PATCH мы можем обновить данные (см. рис. 7).

```
if ($method === 'PUT' && count($urlData) === 1) {
    $fruitId = $urlData[0];
    echo json_encode(array(
        'method' => 'PUT',
        'id' => $fruitId,
        'formData' => $formData
    ));

    return;
}
```

Рисунок 7. Обработчик запроса PUT

Теперь при запросе обновляются данные продукта. Впишем в консоль «curl -X PUT http://test/fruits/20 --data "fruit=apple&price=90" -i» (см. рис. 8).

```
E:\OSPanel\domains\test>curl -X PUT http://test/fruits/20 --data "fruit=apple&price=90" -i
HTTP/1.1 200 OK
Date: Thu, 29 Aug 2019 11:32:48 GMT
Server: Apache
Content-Length: 72
Content-Type: text/html; charset=UTF-8

{"method":"PUT","id":"20","formData":{"fruit":"apple","price":"90"}}
```

Рисунок 8. Результат запроса PUT

Обновлять часть данных мы можем при помощи PATCH. Так же при помощи запроса DELETE можно удалять данные (см. рис.9).

```php
if ($method === 'DELETE' && count($urlData) === 1) {
    $goodId = $urlData[0];
    echo json_encode(array(
        'method' => 'DELETE',
        'id' => $goodId
    ));
    return;
}
```

Рисунок 9. Обработчик запроса DELETE

Теперь при запросе данные продукта удалятся. Впишем в консоль «curl -X DELETE http://test/fruits/20 -i» (см. рис. 10).

```
E:\OSPanel\domains\test>curl -X DELETE http://test/fruits/20 -i
HTTP/1.1 200 OK
Date: Thu, 29 Aug 2019 11:38:06 GMT
Server: Apache
Content-Length: 33
Content-Type: text/html; charset=UTF-8


{"method":"DELETE","id":"20"}
```

Рисунок 10. Результат запроса DELETE

В результаты была разработана REST архитектура для интернет-магазина. В дальнейшем можно привязать базу данных и работать в ней через запросы. Данный шаблон можно использовать как основу в проектирование любого веб-сервиса, где необходима REST архитектура.

Библиографический список

1. Федотова К.В., Хоук А. Обзор вариантов использования REST в современной архитектуре web-приложений//Безопасность городской среды Материалы V Международной научно-практической конференции. Под ред. Е.Ю. Тюменцевой. 2018. С. 410-412.
2. Егунов В.А., Князев А.А. Проектирование оптимизированной архитектуры web-клиента, взаимодействующего с REST API // Известия Волгоградского государственного технического университета. 2018. № 8 (218). С. 79-83.
3. Зотова Ю.А., Котилевец И.Д. Разработка архитектуры rest api для взаимодействия с сервисами приложения // Информационные технологии и математическое моделирование систем 2018 труды международной научно-технической конференции. 2018. С. 61-65.
4. Дроздов С.А., Луканина В.Е. Особенности проектирования серверного и клиентского программного обеспечения web-сайта с использованием rest-архитектуры // Вестник МГУП имени Ивана Федорова. 2016. № 2. С. 74-76.