Жаврук Н. В.

Науковий керівник О. М. Шимон

асистент кафедри прикладної математики та інформатики

Житомирський державний університет ім. І. Франка

ВИКОРИСТАННЯ ТЕХНОЛОГІЇ REST ДЛЯ ПОБУДОВИ ВЕБ-СЕРВІСІВ

Анотація: *У статті розглянуто основні складові технології REST: протокол HTTP, архітектура REST. Описано принципи побудови RESTful API додатків. Наведено приклади коду реалізації таких API.*

Ключові слова: *веб-сервіс, REST, HTTP, RESTful API*

Анотация: *В статье рассмотрены основные составляющие технологии REST: протокол HTTP, архитектура REST. Описаны принципы построения RESTful API приложений. Приведены примеры кода реализации таких API.*

Ключевые слова: *веб-сервис, REST, HTTP, RESTful API*

Abstract: *The article describes the basic components of technology REST: protocol HTTP, architecture REST. The principles of building RESTful API applications. Examples of the implementation of such a code API.*

Keywords: *web-service, REST, HTTP, RESTful API.*

Щохвилини десятки тисяч гігабіт інформації пересилається через Internet. Це здійснюється за допомогою спеціальних програм, таких, як браузери: Internet Explorer, Opera та ін.; таких, як менеджери завантаження: Reget, FlashGet, Gozzila та ін. Інформація також передається через електронну пошту, Internet-телефонію та ін. Роботу цих програм забезпечують спеціальні протоколи. Одним з таких протоколів є загальновідомий протокол HTTP.

Метою статті є розгляд технології REST для побудови RESTful API додатків, яка ґрунтується на використанні протоколу HTTP.

На даному етапі протокол HTTP відіграє дуже важливу роль в передачі інформації. Більшість інформації, яка передається через Internet передається за допомогою HTTP. Раніше цей протокол використовувався лише для передачі інформації невеликих розмірів (текстових файлів, рисунків і т.д.). За передачу більших файлів відповідав протокол FTP (File Transfer Protocol). Зараз по HTTP передаються файли довільних розмірів і він є основним протоколом, який за це відповідає.

HTTP працює відповідно до моделі клієнт/сервер (надання послуг здійснюється спільною роботою двох процесів: на комп’ютері користувача і на комп’ютері – сервері).

HTTP ґрунтується на парадигмі запитів/відповідей. Запитуюча програма (як правило вона називається клієнт) встановлює зв’язок з обслуговуючою програмою-одержувачем (що називається сервер) і надсилає запит серверу в наступній формі:

- метод запиту, URI, версія протоколу,
- заголовки запиту,
- тіло повідомлення, яке може бути відсутнім.

Сервер відповідає повідомленням, що містить рядок статусу (включаючи версію протоколу і код статусу – успіх чи помилка), за яким слідують заголовки відповіді, що включають в себе інформацію про сервер, метаінформацію про зміст відповіді тощо, і, можливо, саме тіло відповіді. Слід зазначити, що одна програма може бути одночасно і клієнтом і сервером. Використання цих термінів у даному тексті відноситься тільки до ролі, що виконує програма протягом даного конкретного сеансу зв'язку, а не до загальних функцій програми [1].

На основі HTTP базується набір принципів проектування, відомий як REST і дозволяє охопити всю потужність шляхом побудови інтерфейсів, що може використовуватися практично з будь-якого пристрою чи операційної системи.

REST – це стиль архітектури програмного забезпечення для побудови розподілених масштабованих веб-сервісів, заснований на маніпулюванні ресурсами і специфікації HTTP. REST ілюструє розвиток Web архітектури, характеризуючи і регулюючи макровзаємодію чотирьох Web компонентів, а саме серверів, мережевих шлюзів, проксі і клієнтів, без застосування обмежень до індивідуальних учасників. Таким чином, REST по суті визначає правильну поведінку учасників [2].

Загальний вигляд використання REST-архітектури показано на Рис.1.

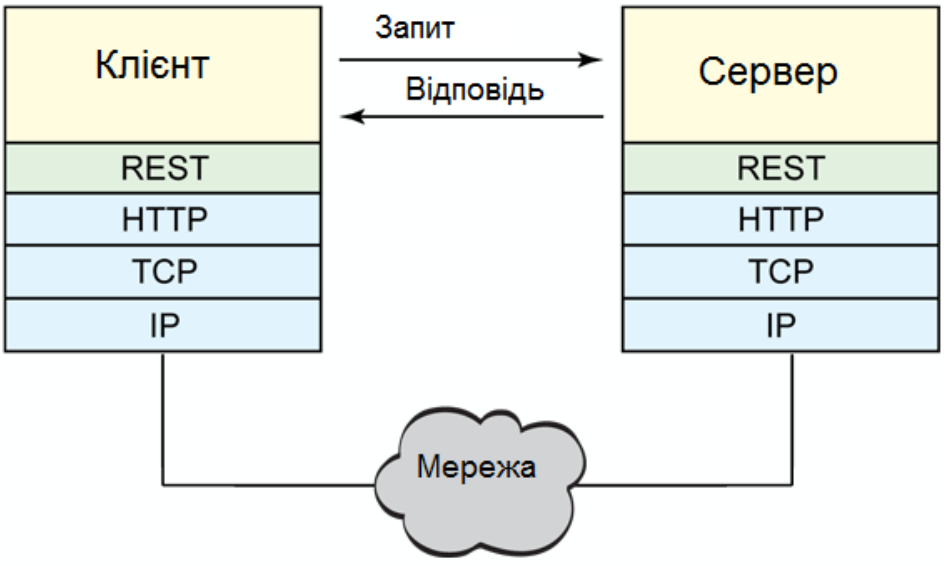

Рис. 1. Багаторівнева архітектура RESTful-взаємодії

В протоколі HTTP описані 8 різних методів для здійсненні запиту до сервера: GET, POST, PUT, DELETE, OPTIONS, HEAD, TRACE, CONNECT. При перегляді веб-сторінок у браузері, використовуються два методи: GET – для передачі запиту на отримання нової веб-сторінки та для передачі даних невеликого розміру з форми, POST – для передачі даних великого розміру або даних, які потрібно приховати.

Загальновживаний термін API означає «програмний інтерфейс додатку». Наприклад, API використовується для опису особливостей програмної бібліотеки, способів взаємодії з деякою програмою тощо. В технології REST API (HTTP API) використовується як спосіб обміну даними веб-додатками. Наприклад, Twitter має API, які дозволяють запитувати твіти у форматі зручному для імпорту у ваші додатки.

Розглянемо приклади деяких HTTP API, які не є RESTful API [3]. Ці API дозволяють проглядати, створювати, редагувати та видаляти деякі віджети:

http://example.com/view_widgets

http://example.com/create_new_widget?name=Widgetizer

http://example.com/update_widget?id=123&name=Foo

http://example.com/delete_widget?id=123

Доступ до таких API може бути здійснена з використанням методу GET, зокрема, отримати відповідь можна ввівши відповідний URI в браузері.

На відміну від розглянутого підходу в технології REST методи HTTP використовуються для передачі команд, які потрібно виконати на сервері. Використовуються методи GET, POST, PUT, DELETE, які часто зрівнюють з CREATE, READ, UPDATE, DELETE (CRUD) операціями маніпулювання базами даних. Команди та параметри передаються у вигляді URI та в тілі повідомлення.

Розглянемо приклади таких запитів: 1) для перегляду одного віджету; 2) для перегляду списку віджетів; 3) створення нового віджету на основі

даних, які передані в тілі повідомлення; 4) оновлення віджету, на основі даних, які передані в тілі повідомлення; 5) видалення деяких даних.

1. GET http://example.com/widgets/123
2. GET http://example.com/widgets
3. POST http://example.com/widgets
   Data:
      name = Foobar
4. PUT http://example.com/widgets/123
   Data:
      name = New name
      color = blue
5. DELETE http://example.com/widgets/123

Розглянуті принципи покладені в основу створення відповідних REST-клієнтів та REST-сервісів. Клієнт можна створити з використання будь-якої мови, яка дозволяє відправляти HTTP-запит та отримувати HTTP-відповідь.

Створення сервіса з RESTful API є складнішим. Але такі сервіси також можна писати на більшості серверних мов програмування, зокрема, на PHP.

Розглянемо деякі елементи реалізації RESTful сервера на мові PHP:

1. Використання методів PUT і DELETE.

В PHP можливо без проблем визначити використовуваний HTTP метод, слід звернутися до змінної системного масиву $_SERVER['REQUEST_METHOD']. У випадку web-браузерів там прописаний або GET, або POST метод. Для клієнтських додатків REST потрібна підтримка PUT, DELETE та інших. Але на жаль в PHP нема $_PUT и $_DELETE, на відміну від $_GET и $_POST:

```
$_PUT = array();
if($_SERVER['REQUEST_METHOD'] == 'PUT') {
  $putdata = file_get_contents('php://input');
```

```
    $exploded = explode('&', $putdata);
    foreach($exploded as $pair) {
      $item = explode('=', $pair);
      if(count($item) == 2) {
        $_PUT[urldecode($item[0])] = urldecode($item[1]);
      }
    }
  }
```

2. Відправка вибіркових HTTP 1.1 заголовків

PHP дозволяє налаштовувати заголовки, які відправляються клієнту. У свою чергу заголовок містить код відповіді від сервера. По замовчуванню PHP відповість кодом 200 OK на успішний запит, на використовування функції die() або на створення нового ресурса. Є два шляхи, щоб налаштувати цю ситуацію:

```
Header ('HTTP/1.1 404 Not Found');
```

Або

```
Header ('Location:http://www.foo.com/bar', true, 201); //201 CREATED
```

Перший варіант достатньо стандартний метод установки коду відповіді. Якщо ж потрібно вказати перенаправлення на ресурс «201 Created» або «301 Moved Permanently», це неважко зробити, вказавши код в третьому параметрі функції header(), як показано в прикладі. Після цього код дещо зміниться, але його буде зручніше читати:

```
header('HTTP/1.1 201 Created');
header('Location:http://www/foo.com/bar`);
```

3. Відправка значущих HTTP заголовків

Суть полягає у використанні прописаних у специфікації заголовків, які є цілком самодостатні, але часто ігноруються більш стереотипними рішеннями.

201 Created — використовується якщо був створений новий ресурс. Повинен застосовуватися із функціоналом направлення на ресурс, наприклад /tech/news, так як сам автоматично не направляється на ресурс.

202 Accepted — На відміну від коду 201, який відсилається після створення ресурсу, код 202 ставить запит в чергу.

204 No Content — у зв'язку з кешуванням і умовним GET запитом (запити з If-Modified-Since/If-None-Match заголовками), дозволяє сказати web-додаткам «контент не змінився, продовжуйте використовувати кешовану версію», без повторної переробки і відравки кеша.

401 Unauthorized — повинен використовуватися при спробі звернутися до ресурсу, який потребує визначений рівень доступу. Використовується у зв'язці з www-authentication.

500 Internal Server Error — набагато краще ніж 200 OK, коли PHP скрипт уже привів до виключення.

Насправді заголовків набагато більше, наведенні вище – лише приклади.

Отже, розробку будь-якого API слід починати з детального опрацювання інтерфейсу. Фактично до початку написання коду на руках повинні бути всі URI вашого API з докладною документацією всіх параметрів кожного доступного для даного URI методу, з усіма повернутими статус кодами і форматами повернутих даних. В ідеалі, цей інтерфейс уже не повинен змінюватися в ході подальшої розробки. Такий підхід в значній мірі полегшує і прискорює роботу над основним кодом API, і дозволяє паралельно писати клієнтське ПЗ вже на самих початкових етапах розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ:

1. В. В. Самсонов, А. Л. Эрохін. Методи та засоби Інтернет-технологій. Х.: СМІТ, 2008, - 264с.

2. М. Тим Джонс. [Електронний ресурс] / . М. Тим Джонс. -Режим доступу: http://www.ibm.com/developerworks/ru/library/os-understand-rest-ruby/ - використання REST – технологій, RUBY.

3. Andrew Havens. Beginners guide to creating a REST API [Електронний ресурс]. – Режим доступу: http://www.andrewhavens.com/posts/20/beginners-guide-to-creating-a-rest-api/