УДК 004.9

ВИКОРИСТАННЯ RESTFUL API ПРИ РОЗРОБЦІ МОДУЛЯ КЕРУВАННЯ ТОВАРНИМИ ЗАПАСАМИ.

Педоренко Владислав, Месюра Володимир

Вінницький національний технічний університет

Анотація

Проведено аналіз використавння RESTful API при розробці модуля керування товарними запасами. Розглянуто базові принципи проектування, схему взаємодії різних частин модуля.

Abstract

Was performed the analysis of the use of the RESTful API in the development of the inventory management module. The basic principles of design, the scheme of interaction of different parts of the module are considered.

Вступ

При розробці модуля керування товарними запасами використовується трирівнева архітектура проектування, тобто будується екосистема, до якої входить: клієнтська частина, серверна частина, яка буде побудована за принципом Restful API та шар даних.

REST визначає ряд архітектурних принципів проектування Web-сервісів, орієнтованих на системні ресурси, включаючи способи обробки і передачі станів ресурсів по HTTP різноманітними клієнтськими додатками, написаними на різних мовах програмування. За останні кілька років REST стала переважаючою моделлю проектування Web-сервісів. Фактично REST зробила настільки великий вплив на Web, що практично витіснила дизайн інтерфейсу, заснований на SOAP і WSDL, через більш простий стиль проектування.

Результати дослідження

REST - це набір принципів і обмежень взаємодії клієнта і сервера в мережі інтернет, який використовує існуючі стандарти (HTTP протокол, стандарт побудови URL, формати даних JSON і XML) в ході взаємодії.

Схема взаємодії клієнтської частини модуля керування товарними запасами з RESTful API наведена на рисунку 1.

Розробляючи модуль керування товарними запасами, REST використовується в чистому вигляді, тобто слідує чотирьом базовим принципам проектування:

– Явне використання HTTP-методів.

– Незбереження стану.

– Надання URI, аналогічних структурі каталогів.

– Передача даних в XML, JavaScript Object Notation (JSON) або в обох форматах.

Однією з ключових характеристик RESTful API при розробці модуля є явне використання HTTP-методів згідно з протоколом, визначеним в RFC 2616. Цей основний принцип проектування REST встановлює однозначну відповідність між операціями create, read, update і delete (CRUD) і HTTP-методами. Згідно з цим, для роботи з ресурсами використовується:

– Для створення ресурсу на сервері використовується POST.

– Для вилучення ресурсу використовується GET.

– Для зміни стану ресурсу або його поновлення використовується PUT.

– Для видалення ресурсу використовується DELETE.

Слідуючи загальноприйнятим підходам, відповідно рекомендаціям REST при явному застосуванні HTTP-методів, використовуються іменники в URI замість дієслів.В RESTful API дієслова POST, GET, PUT і DELETE вже визначені протоколом.

Для реалізації узагальненого інтерфейсу і явного виклику операцій клієнтськими додатками API не визначає додаткові команди або віддалені процедури, наприклад / adduser або / updateuser. Цей загальний принцип застосовано також до тіла HTTP-запиту, який призначений для передачі стану ресурсу, а не імені викликаного віддаленого методу або віддаленої процедури.

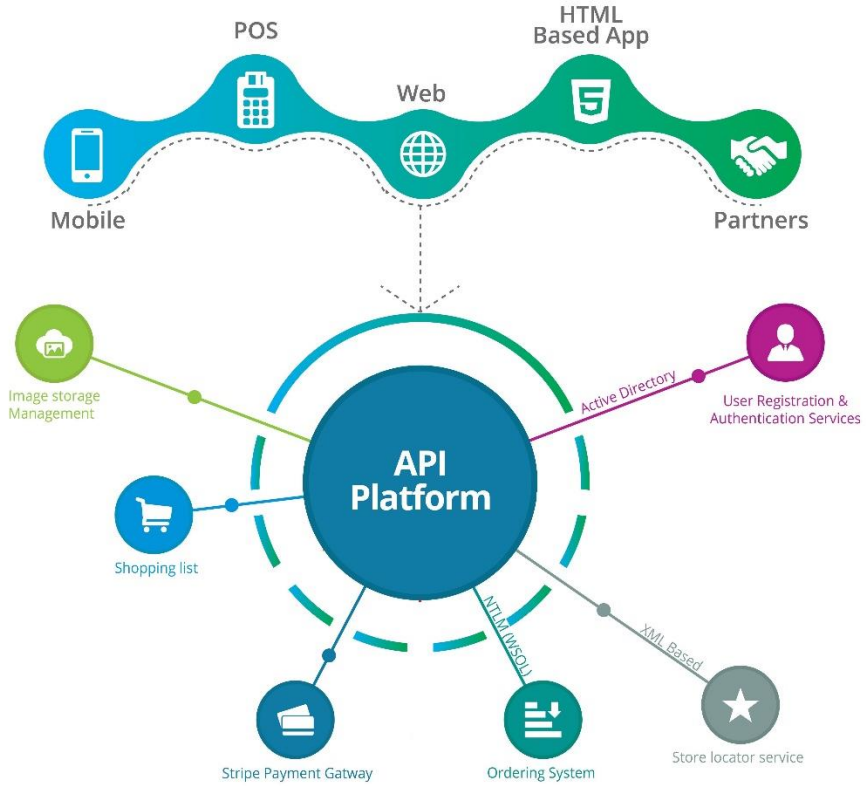

Рисунок 1 – Схема взаємодії клієнтської частини з RESTful API

Для задоволення постійно зростаючих вимог до продуктивності, REST API розробляється масштабованим. Для формування топології сервісів, що дозволять при необхідності перенаправляти запити з одного сервера на інший з метою зменшення загального часу реакції на виклик Web-сервісу, застосовано кластери серверів з можливістю розподілу навантаження і аварійного перемикання на резерв, проксі-сервери і шлюзи. Використовуючи проміжні сервери для поліпшення масштабованості клієнти Web-сервісів REST відправляють повні самодостатні запити, що містять всі необхідні для їх виконання дані, щоб компоненти на проміжних серверах могли перенаправляти, маршрутизувати і розподіляти навантаження без локального збереження стану між запитами.

При обробці повного самодостатнього запиту сервер не витягує стан або контекст програми. Додаток (або клієнт) Web-сервісу REST включає в HTTP-заголовки і в тіло запиту всі параметри, контекст і дані, необхідні серверному компоненту для генерування відповіді.

Для забезпечення функціонування сервісів RESTful API, які не зберігають стан, його розділено на дві сфери відповідальності:

Сервер

– Генерує відповіді, які містять посилання на інші ресурси для навігації додатків по зв'язаних ресурсах. Такі відповіді містять вбудовані посилання. Аналогічним чином при запиті батьківського або контейнерного ресурсу RESTful-відповідь містить посилання на нащадків батьківського елемента або на підлеглі ресурси, щоб зберігають зв'язок з ними.

– Генерує відповіді, що містять інформацію про те, чи підлягають вони кешуванню з метою підвищення продуктивності за рахунок зменшення кількості запитів, що дублюються ресурсів і повної відмови від деяких запитів. Для цього сервер включає в відповідь HTTP-заголовки Cache-Control і Last-Modified (значення даних).

Клієнтська програма

– За заголовком Cache-Control відповіді визначає можливість кешування ресурсу (його локального копіювання). Також клієнтська програма читає заголовок Last-Modified відповіді і повертає значення дати в заголовку If-Modified-Since для відправки на сервер запиту про зміну ресурсу. Такий запит, т.зв. Conditional GET, використовує обидва заголовка. Якщо ресурс з зазначеного часу не змінився, відповіддю сервера є стандартний код 304 (Not Modified) і запитаний ресурс не відправляється. Код відповіді 304 HTTP означає, що клієнтський додаток може спокійно використовувати кешовану локальну копію подання ресурсу в якості самої останньої його версії, фактично опускаючи наступні запити GET до тих пір, поки ресурс не буде змінений.

– Відправляє повні запити, які можуть оброблятися незалежно від інших запитів. Це вимагає від клієнтського додатка використання в повному обсязі HTTP-заголовків, визначених інтерфейсом Web-сервісу, і відправки повних представлень ресурсів в тілі запиту. Клієнтська програма відправляє запити, які практично нічого не знають про попередні запити, про існування сеансу на сервері, про здатність сервера додавати контекст в запит і про стан додатки, що зберігається між запитами.

Така спільна робота клієнтського додатка і сервісу дуже важлива для відмови від збереження стану в Web-сервісах RESTful. Результатом відмови від збереження стану є зростання продуктивності за рахунок зменшення трафіку і мінімізації стану серверного додатка.

URI-адреси REST розроблені щоб бути інтуїтивно зрозумілими. Тому структура URI є простою, передбачуваною і зрозумілою. Одним із способів досягти такого рівня зручності використання – була побудова URI за аналогією зі структурою каталогів. Такого роду URI є ієрархічними, що походить із одного кореневого шляху, розгалуження якого відображають основні функції сервісу.

Подання системних ресурсів через RESTful API - це гнучкий спосіб забезпечення різних додатків даними в стандартному форматі. Це допомагає виконати вимоги до інтеграції, що мають вирішальне значення для створення систем, що дозволяють легко комбінувати дані, а також для побудови на основі набору базових RESTful-сервісів більших конструкцій.

Список використаних джерел:

1. Leonard Richardson, Sam Ruby, RESTful Web Services - O'Reilly Media, 2008. – 235 с.

2. Педоренко, В.Ю. Модуль керування товарними запасами, клієнтська частина [Текст]/В.Ю. Педоренко, В.І. Месюра // XLVII Науково-технічна конференція ф-ту інформ. технологій та комп. інженерії : Тез. доп. – Вінниця, 2018. – с. 2