УДК 004.03

Г.О. Горбачов
М. О. Шевчук
О. М. Ткаченко

АНАЛІЗ ПІДХОДІВ ДО ПРОЕКТУВАННЯ КЛІЄНТ-СЕРВЕРНИХ СЕРВІСІВ

Вінницький національний технічний університет

Анотація

За результатами проведеного аналізу виявлено недоліки та переваги основних підходів до проектування клієнт-серверних сервісів та визначено доцільність розподілу відповідальності між компонентами програмного забезпечення.

Ключові слова: веб-сервіс, клієнт-серверна архітектура, stateless сервіс, stateful сервіс, REST, SOAP.

Abstract

According to the results of the analysis revealed the shortcomings and advantages of the basic approaches to designing client-server services and determined the expediency of the distribution of responsibilities between the components of the software.

Keywords: web-service, client-server architecture, stateless service, stateful service, REST, SOAP.

Актуальність обраної тематики обумовлена необхідністю аналізу та на його основі отримання певних висновків про використання шаблонів проектування клієнт-серверних сервісів та визначити доцільність розподілу відповідальності між компонентами програмного забезпечення. Неправильно вибрана архітектура сервісу сильно обмежує можливості розширення та ускладнює підтримку сервісу. Зміна архітектури вже готового сервісу вимагає багато ресурсів, тому важливо одразу обрати правильний для конкретної задачі варіант.

Веб-сервіс, як правило поділяється на серверну та клієнтську частини, де серверна частина містить бізнес логіку та оперує над даними, а клієнтська частина створює представлення для зручного використання сервісу кінцевим користувачем. Такий підхід дає можливість чітко розмежувати відповідальність між компонентами, що робить їх максимально незалежними один від одного. Комунікація між компонентами відбувається за допомогою мережевого з’єднання. Зазвичай веб-сервіси працюють поверх інтернет протоколу http або протоколів вищого рівня.

В контексті розробки веб-сервісів існують дві основні концепції: stateful та stateless. Stateful – це підхід зі збереженням стану. Його сенс полягає в тому, щоб опиратися на стан об’єкта у часі і в залежності від нього та певних вхідних даних змінювати результат виконання. Як приклад можна використати традиційний FTP-сервер. Кожна зміна стану користувача, наприклад запис про активний каталог, зберігається на сервері як стан клієнта. Кожна зміна, внесена на сервер, реєструється як зміна стану, і коли користувач відключається, його стан відповідно змінюється на від’єднаний. При такому підході можуть виникати деякі проблеми. Перш за все з’являється велика кількість незакінчених сесій та транзакцій. В деяких ситуаціях може бути незрозумілим як довго сервіс має залишати сеанс відкритим, або як йому дізнатися чи клієнт від’єднаний. Хоча існують обхідні шляхи для всіх цих проблем, найчастіше, збереження стану корисно тільки якщо самі функції залежать від стану. Більшість користувачів можуть взаємодіяти з веб-сервісом різними способами, і тому збереження стану сервера не залежить від програми-клієнта, так як якщо клієнт не може реалізувати функціонал сервісу, то і в збереженні стану немає ніякої необхідності.

Stateless підхід – це зовсім інша концепція. Вона базується на відсутності будь-якого стану, що дозволяє уникати проблем stateful. Це надає можливість обробляти запити незалежно від стану системи ґрунтуючись лише на вхідних даних та наборі інструкцій по їх обробці. Концепція stateless є фундаментальним аспектом сучасного інтернету. Наприклад при перегляді новин використовується протокол HTTP для підключення без урахування стану з використанням повідомлень, які аналізуються та обробляються ізольовано один від одного та від вашого стану.

На базі цих концепцій було створено дві реалізації: SOAP та REST. SOAP це стандартизований протокол, що передає повідомлення с використанням протоколів HTTP, TCP, UDP, SMTP та інших. Специфікації SOAP є офіційними веб-стандартами що підтримуються та розробляються Консорціумом World Wide Web. Перевагами використання SOAP є чіткі правила використання, розширені функції безпеки, включаючи ACID принципи та авторизацію. Але з іншої сторони використовуючи SOAP сильно ускладняється архітектура, що в свою чергу знижує можливість розширення сервісу. REST був створений для вирішення проблем SOAP, відмовляючись від строгих специфікацій сервіс стає більш гнучким, адже розробники можуть самостійно реалізовувати низькорівневі деталі, або використовувати рішення створені кимось іншим. Можна сказати що REST був спеціально розроблений як функціонально не маючий стану. Вся концепція передачі репрезентативного стану залежить від ідеї передачі всіх даних для обробки запиту таким чином, щоб всі необхідні дані уже містилися в самому запиті.

При виборі технології що буде використовуватись для написання веб сервісу варто визначитися з декількома питаннями, а саме які масштаби сервісу та яка його орієнтована аудиторія. Якщо сервіс буде використовуватись як менеджер для збереження великої кількості даних від необмеженої кількості користувачів варто використовувати REST, адже таким чином пропускна спроможність серверу буде більшою що дозволить витримувати більші навантаження. Якщо сервіс орієнтується для використання у корпоративній мережі, містить складну бізнес логіку або ресурсоємні обчислення, потребує можливості використання різних протоколів то варто використовувати SOAP. У більшості випадків можливості SOAP є надмірними, і варто використовувати більш гнучку REST архітектуру.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Richardson L., Ruby S. RESTful Web Services. — O'Reilly Media, 2007. — 454 c.
2. Miller E. RESTful Web Service Architecture. — indle Edition, First Edition 2011— 19 c.
3. Defining Stateful vs Stateless Web Services. [Електронний ресурс]. — Режим доступу: https://nordicapis.com/defining-stateful-vs-stateless-web-services/
4. The What, Why, and How of a Microservices Architecture [Електронний ресурс]. — Режим доступу: https://medium.com/hashmapinc/the-what-why-and-how-of-a-microservices-architecture-4179579423a9

Горбачов Геннадій Олександрович — студент кафедри комп'ютерної інженерії ВНТУ, Вінницький національний технічний університет, м. Вінниця, e-mail: genagorbachov1@gmail.com

Шевчук Максим Олегович— студент кафедри комп'ютерної інженерії ВНТУ, Вінницький національний технічний університет, м. Вінниця, e-mail: m.shevchuk.o@gmail.com

Ткаченко Олександр Миколайович — професор кафедри комп'ютерних наук, Вінницький національний технічний університет, м. Вінниця, e-mail: alextk1960@gmail.com

Horbachov Hennadii Oleksandrovych — student of the Computer Techniques Chair, Vinnytsia National Technical University, Vinnytsia, e-mail: genagorbachov1@gmail.com

Shevchuk Maksym Olehovych — student of the Computer Techniques Chair, Vinnytsia National Technical University, Vinnytsia, e-mail: m.shevchuk.o@gmail.com

Tkachenko Oleksandr Mykolaiovych — Professor of the Computer Techniques Chair, Vinnytsia National Technical University, Vinnytsia, e-mail: alextk1960@gmail.com