АРХІТЕКТУРА ТА КОМПОНЕНТИ КОМП'ЮТЕРНИХ СИСТЕМ

УДК 681.3, 621.3

І. Пастернак
Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин

ПАРАМЕТРИЗОВАНІ МЕРЕЖНІ ІНТЕРФЕЙСИ

Запропоновано варіант клієнт-серверної взаємодії на основі віддалених об'єктів у глобальній мережі, а також запропоновано використання та класифікацію шаблонів під час побудови та параметризації мережних інтерфейсів.

Ключові слова: віддалений об'єкт, шаблон, клієнт-серверна взаємодія, параметризований мережний інтерфейс.

The variant of client-server interaction based on distant objects in the global network. Also proposed use and classification of patterns in the construction and parameterization of network interfaces.

Key words: remote object, pattern, client-server interaction parameter network interface.

Вступ

Використання віддалених об'єктів в мережі за взаємодії клієнт-сервер дуже складно, особливо якщо їх потрібно використовувати повторно. Необхідно підібрати відповідні віддалені об'єкти, віднести їх до різних класів, зберігаючи ступінь деталізації, визначити параметризацію інтерфейсів і ієрархію наслідування і встановити відносини між параметрами. Параметризований інтерфейс повинен, з одного боку, відповідати задачі, яка розв'язується, а з іншого – бути загальним, щоб вдалося врахувати всі вимоги, які можуть виникнути в майбутньому. Розробники скажуть вам, що забезпечити «правильний», тобто достатньою мірою гнучкий та придатний для повторного використання параметризований мережний інтерфейс, з першого разу важко, якщо взагалі можливо. Перед тим як вважати мету досягнутою, вони зазвичай намагаються випробувати знайдене рішення на декількох задачах, і кожного разу модифікують його завдяки параметризації мережного інтерфейсу.

І все ж таки розробникам вдається створити параметризований мережний інтерфейс системи. Водночас користувачі вражені кількістю можливих варіантів і часто повертаються до звиклих не об'єктно-орієнтованих методик. Проходить багато часу перед тим, як стає зрозуміло, що краще використовувати об'єктно-орієнтований параметризований мережний інтерфейс через можливість зміни параметрів, які в результаті можуть змінити вигляд цього інтерфейсу [1, 3]. Завдяки цьому розробнику зрозуміло, що не потрібно розв'язувати кожну нову задачу з самого початку. Замість цього він намагається повторно скористатися тими рішеннями, котрі виявилися вдалими в минулому. Відшукавши одне рішення відразу, він використовуватиме його знову і знову. В багатьох об'єктно-орієнтованих системах ви зустрінете шаблони, які повторюються і які складаються з класів і взаємодіючих об'єктів. Саме враховуючи їх, можна параметризувати мережний інтерфейс.

З використанням шаблонів розробники програмного забезпечення вирішують конкретні задачі, внаслідок чого об'єктно-орієнтований вигляд параметризованого мережного інтерфейсу стає більш

гнучким, ефективним, і його можна застосовувати повторно. Розробник програмного забезпечення, який знайомий з шаблонами, може зразу використовувати їх для розв’язання нової задачі, не намагаючись кожного разу винайти щось нове і знову параметризувати мережний інтерфейс.

Огляд літературних джерел

За рахунок параметризації мережного інтерфейсу за клієнт-серверної взаємодії клієнт з сервером можуть обмінюватися віддаленими об’єктами в глобальній мережі. Розглянемо кроки, необхідні, аби дозволити клієнту отримати доступ до об'єкта на віддаленому сервері:

- Створити TCP або HTTP-з'єднання між клієнтом і сервером через параметризований мережний інтерфейс.
- Вибрати повідомлення, які відправлені від клієнта до сервера та відформатувати їх.
- Реєструвати типи, які будуть доступні у віддаленому режимі.
- Створити віддалений об'єкт та активувати його з сервера або клієнта.

Ви не повинні розуміти, як реалізувати параметризований мережний інтерфейс, чи знати, як працюють протоколи TCP, HTTP, або знати порти – треба просто вказати тип з’єднання і який порт використовувати, що можна задати автоматично. Якщо вибрано протокол http, для з’єднання необхідно використовувати Simple Object Access Protocol (SOAP) в бінарному форматі, який використовує TCP. За замовчуванням SOAP використовується тоді, коли вибраний протокол HTTP і TCP тоді реалізується бінарне форматування. Хоча це і не так ефективно, як у двійковому форматі, комбінація SOAP і HTTP стала стандартом де-факто для передавання даних через брандмауери, який використовується для реалізації параметризованого мережного інтерфейсу. Двійковий формат рекомендується у випадках, коли брандмауер не є проблемою. Віддалені об’єкти створюються з класів. Вони знаходяться на віддаленому хості, щоб реалізувати канал для з’єднання клієнта з сервером та передачу віддаленого об’єкта, до яких вони надають доступ [2].

Додаток може підтримувати кілька режимів ввімкнення і кілька віддалених об'єктів. Клієнт вказує на об'єкт, який він хоче отримати, маючи доступ до сервера і використовувати або клієнт, або сервер активації режиму. А також клієнт вибирає метод реєстрації і передає декілька його параметрів, які визначають адресу сервера і тип (клас), який буде доступний. Сервер реєструє типи з’єднання і порти, які він зробить доступними для клієнтів. Сьогодні є чимало прикладів коду, метою якого є презентація прототипів, які можна використовувати для реалізації широкого спектра віддалених додатків. Наведено приклади сервера, який отримує повідомлення і відправляє їх клієнту за запитом. Коли клієнт намагається викликати метод віддаленого об'єкта, його виклик проходить крізь кілька шарів на боці клієнта. Перший з них являє собою проксі-абстрактний клас, який має такий самий параметризований мережний інтерфейс, як віддалений об'єкт, який він представляє. Це підтверджує, що кількість і тип аргументів у виклику вірні, упаковує запит в повідомленні і передає його клієнту через канал. Канал відповідає за транспортування запиту на віддалений об'єкт. Канал складається з приймача, що серіалізує запит в потік і транспортує клієнту, який фактично передає запит в порт на сервері.

Сервер, з іншого боку, вказує, які об'єкти він хоче зробити доступними для видалених клієнтів і активізує режим, необхідний для доступу до них. Це робиться за допомогою механізму, відомого як тип реєстрації. Процес реєстрації відбувається на сервері і клієнті. У своїй найпростішій формі реєстрація на хост полягає у створенні екземпляра об'єкта каналу та його реєстрації передаванням як параметра статичного методу відповідного класу. Цей приклад реєструє канал HTTP на порт 3200. Є кілька правил, щоб реєструвати канал для взаємодії клієнта з сервером:

- І сервер, і клієнт можуть зареєструвати кілька каналів, однак клієнт повинен зареєструвати канал, який відповідає одному хосту.
- Кілька каналів не можуть використовувати той самий порт.
- За замовчуванням, HTTP і TCP протоколам присвоюються імена HTTP і TCP відповідно. За спроби зареєструвати кілька HTTP або TCP-канали з іменем за замовчуванням ви отримаєте виняток. Способом вирішення цієї проблеми є створення каналів за допомогою конструктора, який приймає параметр «ім'я» каналу.

Альтернативою є вкладенний канал, протокол якого дозволяє йому бути зазначеним у файлах конфігурації, пов'язаний з клієнтом і хост-вузлом. Коли об'єкт стану віддаленого об’єкта за значенням отримує копію об'єкта у власному домені додатку, він може працювати з об'єктом на місцевому рівні і не потребує проксі-сервера. Такий підхід не є ефективнішим, ніж стан віддаленого об’єкта за посиланням, де всі запити є на віддаленому об'єкті. Тим не менш виклик віддалених об'єктів, які призначені для запуску на клієнтських комп’ютерах, реалізувати простіше, ніж на сервері, і це може зменшити витрати часу клієнтських запитів.

Постановка задачі

Реалізувати клієнт-серверну взаємодію за допомогою параметризованого мережного інтерфейсу на основі шаблонів.

Основні результати досліджень

Для ефективної клієнт-серверної взаємодії в глобальній мережі використовують параметризований мережний інтерфейс, тобто мережний інтерфейс, який параметризується і наслідується за допомогою класів з використанням шаблонів (рис. 1).

Рис. 1. Схематичне зображення клієнт-серверної взаємодії з використанням параметризованого мережного інтерфейсу

Коли клієнт надсилає запит на віддалений об’єкт на сервері, доступ до нього він отримує через мережний інтерфейс. Мережний інтерфейс, своєю чергою, можна параметризувати, щоб спростити взаємодію клієнта з сервером через віддалені об’єкти. А саме змінюючи один параметр з використанням параметризованого мережного інтерфейсу можна вирішувати різні поставлені завдання, які потрібно застосовувати під час з’єднання клієнта з сервером в глобальній мережі. Для ефективної параметризації і наслідування мережного інтерфейсу використовують шаблони. Клас взаємодії і форматування інстанціює шаблони, які переводить у певний формат.

Програмуючи параметризований мережний інтерфейс, важливо знати, що шаблони відрізняються від класів за ступенем деталізації і рівнем абстракції і повинні бути якимось чином організовані у параметризований мережний інтерфейс. Будемо класифікувати шаблони за двома критеріями (див. таблицю). Першим критерієм буде призначення шаблону. У зв’язку з цим виділяють породжувальні шаблони, структурні шаблони і шаблони поведінки. Перші пов’язані з процесом створення об’єктів. Другі мають відношення до композиції об’єктів та класів. Шаблони поведінки характеризують те, як класи чи об’єкти взаємодіють між собою.

Класифікація шаблонів

Мета\Рівень	Породжувальні шаблони	Структурні шаблони	Шаблони поведінки
Клас	Фабричний метод	Адаптер (класса)	Інтерпретатор
			Шаблонний метод
Віддалений об’єкт	Абстрактна фабрика	Адаптер (об’єкта)	Ітератор
	Одинак	Декоратор	Команда
	Прототип	Заступник	Спостерігач
	Будівник	Компонувальник	Відвідувач
		Міст	Посередник
		Пристосованець	Стан
		Фасад	Стратегія
			Хранитель
			Ланцюжок зобов’язань

Другий критерій – рівень – говорить про те, для чого застосовуються шаблони: до віддалених об’єктів чи до класів. Шаблони рівня класів описують відношення між класами та підкласами. Такі відношення виражаються за допомогою наслідування, тому вони статичні, тобто зафіксовані на етапі компіляції. Шаблони рівня об’єктів описують відношення між об’єктами, котрі можуть змінюватися під час виконання і тому динамічніші. Майже усі шаблони якоюсь мірою використовують наслідування [5]. Тому до категорії «шаблони класів» віднесено тільки ті, які сфокусовані лише на відношеннях між класами. Зверніть увагу: більшість шаблонів діють на рівні віддалених об’єктів.

Породжувальні шаблони класів частково делегують відповідальність за створення віддалених об’єктів своїм підкласам, своєю чергою, породжувальні шаблони віддалених об’єктів передають відповідальність іншому віддаленому об’єкту. Структурні шаблони класів використовують наслідування для вкладеності класів, тоді як структурні шаблони віддалених об’єктів описують способи складання віддалених об’єктів з частин. Поведінкові шаблони класів використовують наслідування для опису алгоритмів і потоку управління, а поведінкові шаблони віддалених об’єктів як об’єкти, які належать деякій групі, разом функціонують і виконують задачу, котра жодному окремому віддаленому об’єкту не під силу.

Взаємодію клієнта з сервером через параметризований мережний інтерфейс на основі шаблонів можна реалізувати через:

- віддалений об’єкт ;
- клієнт активованих об’єктів;
- сервер активованих об’єктів.

Стан віддаленого об’єкта за посиланням використовується, коли клієнт робить запит на об'єкт, який працює на віддаленому сервері. Віддалені об'єкти отримують стан і повинні наслідувати клас. Активізують режим віддаленого об’єкта за одним з двох варіантів: клієнт активованих об'єктів і сервер активованих об'єктів, також зазвичай називають відомі віддалені об'єкти. Це залежить від клієнта: він обирає, який з варіантів використовувати. Якщо клієнт вибирає сервер активованих об'єктів, то підтверджує використання активованого сервером одного запиту або активованого сервером одного з віддалених об'єктів. Вибір активації режиму впливає на загальний вигляд мережного інтерфейсу, продуктивність і масштабованість віддаленого додатку. Він визначає, коли віддалені об'єкти створюються, скільки цих об'єктів створено, як керувати їхнім життєвим циклом і чи підтримуватиме стан віддалених об'єктів інформацію, яку вони містять. На рис. 2 показано взаємодію клієнта з сервером через параметризований мережний інтерфейс за допомогою віддаленого об’єкта.

Коли клієнт намагається спілкуватися з віддаленим об'єктом через параметризований мережний інтерфейс, його посиланням на віддалений об'єкт насправді займається посередник, відомий як сервер активованих об’єктів. Є два типи активації режиму віддалених об’єктів: сервери активованих об’єктів, коли клієнт зв'язується з віддаленими об’єктами напряму, і реальний клієнт активованих об’єктів, коли клієнт приймає запит і пересилає його на віддалений об'єкт [4]. Сервер активованих об’єктів активізує

параметризований мережний інтерфейс для клієнта, який збігається з класом. Активізовані об’єкти переконуються, що всі клієнтські запити відповідають заданому цільовому методу, тобто тип і кількість параметрів збігаються. Розробник відповідає за забезпечення метаданими, які визначають клас віддаленого об’єкта, доступні під час компіляції та виконання.

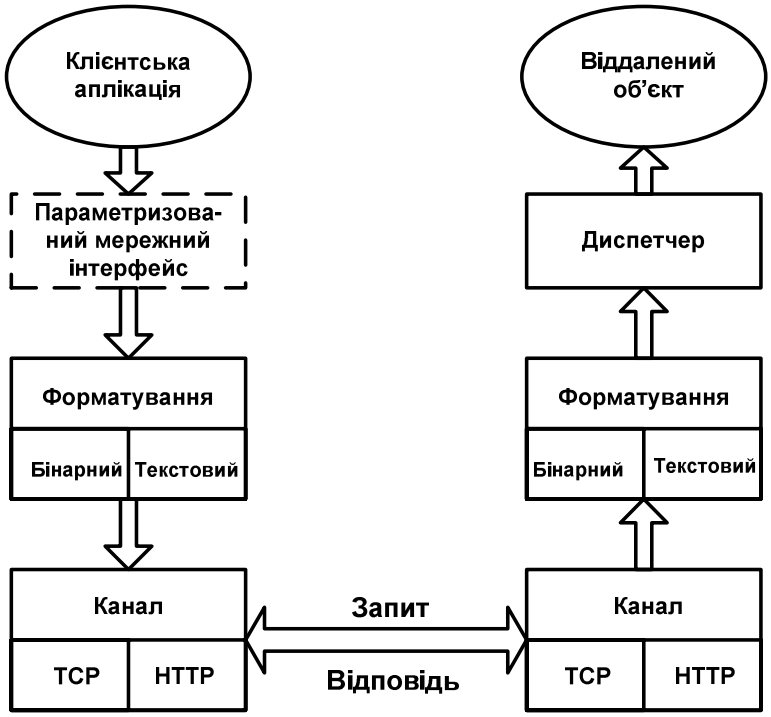

Рис. 2. Блок-схема взаємодії клієнта з сервером через параметризований мережний інтерфейс за допомогою віддаленого об’єкта

Найпростіший спосіб полягає в наданні клієнту копії віддаленого об’єкта на сервері, що містить клас. Після того сервер активованих об’єктів упаковує запит у повідомлення як об'єкт класу, який реалізує параметризований мережний інтерфейс. Повідомлення об'єкта класу передається як параметр для виклику реальних методів, які передають його каналом мережного з’єднання. Там форматується об'єкт, серіалізуются повідомлення і передає його в об'єкт каналом мережного з’єднання, що фізично відправляє повідомлення на віддалений об'єкт.

Використання і поведінка клієнта активізованого об'єкта нагадують, що об'єкт створений локально. Обидва вони можуть бути створені за допомогою нового оператора, як можна використати параметризовані конструктори на додаток за замовчуванням і як зберегти інформацію про стан властивостей або полів. Як показано на рис. 3, вони відрізняються тим, що сервер активованого об'єкта працює на хості в окремому домені програми і називається проксі-сервером.

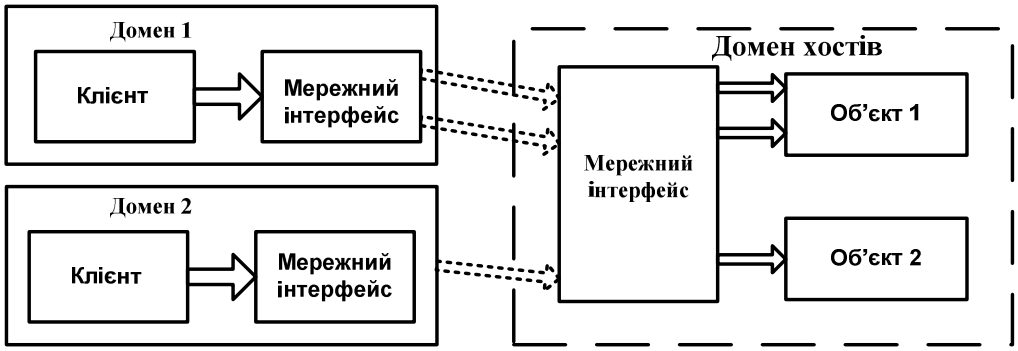

Рис. 3. Спрощена схема вигляду клієнта активованих об'єктів, коли клієнт зберігає контроль над об'єктом

Той факт, що віддалений об'єкт знаходиться в іншому домені, означає, що він просто існує і може бути знищений, навіть якщо клієнт використовує його, як і раніше. Домен обробляє цю потенційну проблему, призначивши оренди кожного віддаленого об'єкта, який може бути використаний, щоб зберегти його активним. Сервер активізованих об’єктів може бути реалізований як з одного виклику віддаленого об'єкта, так і один віддалений об'єкт створюється для кожного запиту з використанням параметризованого мережного інтерфейсу . У цьому випадку якнайкраще підходить для спільного використання одного ресурсу або спільних операцій між декількома користувачами. Одиничний режим використовує виклик, коли клієнти повинні виконати за відносно короткий час декілька операцій на сервері, який не вимагає збереження інформації про стан від одного виклику до іншого. Такий підхід є найбільш масштабованим рішенням і має додаткову перевагу, працює добре в середовищі, яке використовує балансування навантаження на прямі виклики на кілька серверів. Єдиний віддалений об'єкт використовується для обробки всіх викликів в основі вигляду параметризованого мережного інтерфейсу (рис. 4).

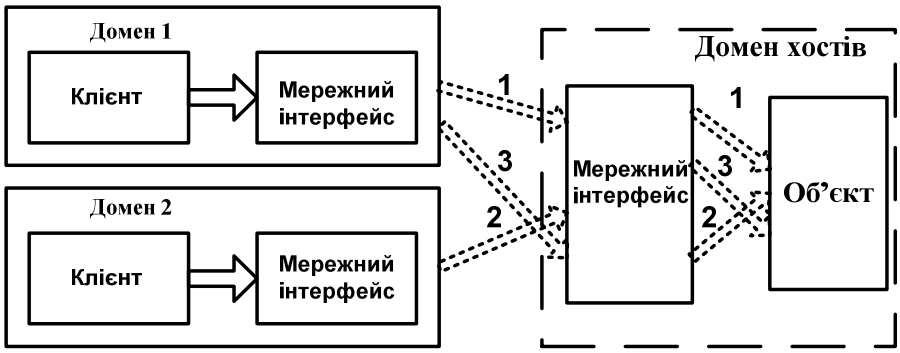

Рис. 4. Спрощена схема вигляду сервера, активованого єдиним об'єктом: один об'єкт обробляє всі запити

Сервер створює віддалений об'єкт, коли перший клієнт намагається отримати до нього доступ, а не коли він намагається створити його. Тому що віддалений об'єкт створюється тільки один раз зусиллями інших клієнтів; замість цього вони всі дали посилання на той самий об'єкт через параметризований мережний інтерфейс. Кожного разу, коли клієнт викликає віддалений об'єкт, виділяється один новий потік з пулу потоків. З цієї причини за це відповідає розробник. Це також обмежує масштабованість, тому що, як правило, лише кінцеве число доступних потоків.

У режимі активації запитів сервер створює новий віддалений об'єкт кожного разу, коли зроблено запит на нього. Після запиту він буде оброблений, віддалений об'єкт буде вимкнено. Кілька запитів обробляють, коли перший запит завершено, і віддалений об'єкт, створений для нього, буде видалений на момент, коли клієнт, який зробив цей запит, посилається на нульове значення (рис. 5).

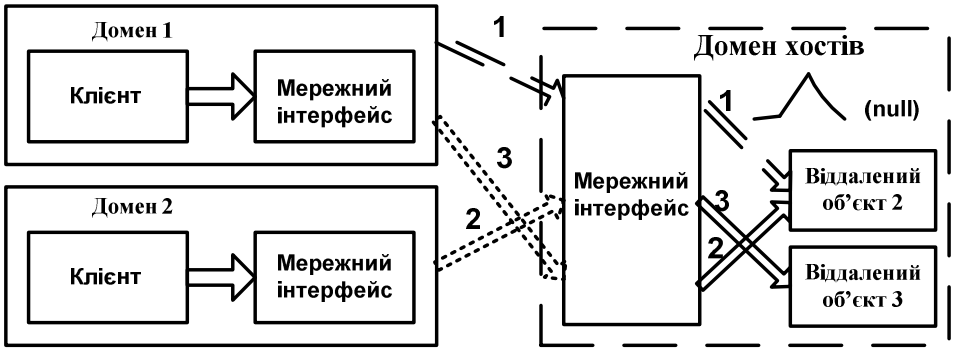

Рис. 5. Спрощена схема вигляду сервера, активованого одним запитом: один об'єкт створюється для кожного запиту

Другий виклик виходить від іншого клієнта і в результаті створення другого об'єкта сервера. Нарешті, перший клієнт робить свій другий запит і тоді буде створено третій віддалений об’єкт. Перевага активізації запиту в тому, що ресурси є доступні, як тільки запит завершений [6]. Це відрізняється від клієнт-активізованих запитів, коли клієнт проводить на ресурси, поки він не закінчив використання віддаленого об'єкта через параметризований мережний інтерфейс. Недолік одного запиту в тому, що за своєю природою не підтримують інформацію про стан між запитами на віддалений об’єкт. Якщо ви хочете скористатися одним масштабованим запитом, але вимагається, щоб інформація про попередні запити збереглася, можна створити сервер для підтримки своєї інформації про стан у файл або базу даних.

Висновки

Було реалізовано клієнт-серверну взаємодію з використанням параметризованого мережного інтерфейсу. В основу цієї реалізації було покладено параметризацію і наслідування шаблонів за рахунок класів. Також було проаналізовано переваги використання шаблонів: вони дають змогу поєднувати досвід експертів і зробити його доступним розробникам; імена шаблонів складають словник, котрий допомагає розробникам краще розуміти один одного; якщо в документації системи вказано, які шаблони в ній використано, то це дає змогу читачу швидше зрозуміти систему; шаблони спрощують реструктуризацію системи незалежно від того, чи використовувались паттерни під час його проектування.

1. Гамм Э., Хелм Р., Джонсон Р., Влиссидес Д. Приёмы объектно-ориентированного проектирования. Паттерны проектирования. – М., 2007. – 366 с. 2. Влиссидес Дж. Применение шаблонов проектирования. – М., 2003. – 130 с. 3. Беркович В., Чудин А. Практика применения паттернов проектирования. – 2006 р. [Електронний ресурс]. – Режим доступу http://rsdn.ru/. 4. ANSI/NISO Z39.50-1995. Information Retrieval (Z39.50): Appli-cation Service Definition and Protocol Specification. Z39.50 Maintenance Agency Offical Text for Z39.50-1995, July 1995. 5. Dave Raggett, Arnaud Le Hors, Ian Jacobs, editors. HTML 4.01 Specification. World Wide Web Consortium, 1999. 6. Федоров А. Платформы и средства создания Web-сервисов // Компьютер-Пресс. – 2002. – № 6. – С. 38–53. 6. Эммерих В. Конструирование распределенных объектов. Методы и средства программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft/COM и Java/RMI: Пер. с англ. – М.: Мир, 2002. – 510 с.