УДК 004.75

А.А. Поляков, В.М. Федорченко, О.П. Іщенко

Харківський національний економічний університет, Харків

ОСОБЛИВОСТІ ВИКОРИСТАННЯ ПРОТОКОЛУ ODATA У РОЗПОДІЛЕНИХ ДОДАТКАХ З ВИКОРИСТАННЯМ МОБІЛЬНИХ ПРИСТРОЇВ

Сучасні підхід використання мобільних пристроїв у корпоративній інфраструктурі потребує доступу до великого обсягу даних,але не великі розміри пам'яті на пристрої, актуальність та конфіденційність даних потребують централізованого зберігання. Існування різних типів баз даних до яких необхідний доступ потребує проміжного шару відображення моделі даних. Механізми протоколу доступу до даних OData дозволяє здійснювати ремапинг даних, а також надавати вибірковій доступ до даних.

Ключові слова: *розподілена обробка даних, REST, ремапинг даних, протокол OData.*

Вступ

При розробці програмних додатків, не зважаючи на їх розмір, завжди основним правилом проектування є відділення моделі даних та бізнес-логіки від компонентів відображення. У процесі еволюції патернів проектування програмного забезпечення та з виникненням великих веб-орієнтованих додатків, склалася така ситуація, що модель та представлення додатку не знаходяться в одному коді, а вже «рознесені географічно», причому модель даних є складною системою сутностей. Інженери з усього світу шукають та розроблюють нові архітектурні рішення, які б дали змогу правильно побудувати такі додатки.

При розробці розподілених , сервіс - орієнтованих додатків у багатьох випадках використовують архітектурний стиль проектування REST. REST – це ресурс-орієнтований підхід. Додаток може взаємодіяти з ресурсами, знаючи дві речі: ідентифікатор ресурсу і необхідну дію . Йому не потрібно знати, чи присутній кеш, проксі-сервер, шлюз, міжмережевий екран, тунель або що-небудь ще між ним і сервером, що володіє реальною інформацією. Найвідомішою системою, побудованою переважно за архітектурою REST, є сучасна Всесвітня павутина

Якщо проаналізувати великі розподілені системи, такі як Facebook, Google, Amazon та інші, то 80 % сервісів як раз розроблені у REST стилі, інші 20% у SOAP стилі. Архітектурний стиль REST забезпечив собі таку популярність при розробці розподілених додатків завдяки наступним перевагам:

- масштабованість окремих компонентів та їх взаємодії;

- клієнти відокремлені від сервера єдиним інтерфейсом. Це розділення відповідальності означає, що клієнти не відповідають за сховище даних, що є внутрішнім для кожного сервера, переносимість коду клієнта поліпшується;

- взаємодія клієнт-сервера обмежується відсутністю збереження контексту клієнта на сервері між запитами. Кожен запит від будь-якого клієнта містить всю інформацію, необхідну для його обслуговування, а будь-який стан сесії зберігається в клієнті. Сервер може бути зі станом; це обмеження вимагає, щоб стан на стороні сервера було адресоване через URL як ресурс. Це не тільки робить сервери більш видимими для моніторингу, а й робить їх більш надійними в разі часткової відмови мережі;

- можливість кешування. Клієнти можуть кешувати відповіді;

- багаторівневість системи;

- легкість у розгортанні сервера;

- єдиний інтерфейс між клієнтами і серверами спрощує і розділяє архітектуру, дозволяючи кожній частині розвиватися самостійно;

Архітектура REST є доволі молодою, вона була розроблена, описана і популяризована у 2000 році Роєм Філдінгом (Roy Fielding), одним із творців протоколу HTTP.

У 2010 році корпорація Майкрософт розробила протокол OData, який базується та є своєрідним розширенням REST. Open Data Protocol (OData) є веб-протокол для запитів і оновлення даних, що дає можливість розблокувати дані і звільнити їх від посиленої інкапсуляції, що існує в сучасних додатках. OData робить це, застосовуючи і спираючись на веб-технології, такі як HTTP, Atom Publishing Protocol (AtomPub) і JSON, щоб забезпечити доступ до інформації з різних додатків, сервісів і магазинів. Протокол базується та є одночасно розширенням протоколу AtomPub. OData може використовуватися для виявлення і доступу до інформації з різних джерел, включаючи реляційні бази даних, файлові системи, системи управління контентом і традиційні веб-сайти. OData використовує основні парадигми Web - він робить глибоку прихильність до URI, для ідентифікації ресурсів і надає єдиний інтерфейс для взаємодії з цими ресурсами (як в Інтернеті). Ця прихильність основним принципам Web дозволяє OData включити новий рівень інтеграції даних і взаємодія з широкого кола клієнтів, серверів, сервісів та інструментів.

Модель використання протоколу OData

Основна проблема, яку повинен вирішувати OData – це створення єдиного інтерфейсу для доступу до різноманітних джерел даних. Є багато можливих джерел даних. Додатки збору та зберігання інформації в базах даних, організації зберігання даних в хмарі, багато фірм роблять бізнес на продажі даних. І так само, як є багато джерел даних, існує безліч можливих клієнтів: веб-браузери, програми для мобільних пристроїв, інструменти бізнес - аналізу та багато іншого. Тому постає проблема організації доступу кожного з клієнтів до джерел даних. А на поточний час це є велика проблема, яка потребує багато ресурсів. Визначення загального підходу має набагато більше сенсу. Все що потрібно, це угода на шляху до моделі даних і протокол для доступу до цих даних - реалізація може відрізнятися. А якщо врахувати, Web-орієнтований світ в якому ми живемо, це мало б сенс OData визначає абстрактну модель даних і протокол , що описують джерело даних. Загальна схема представлена на рис. 1.

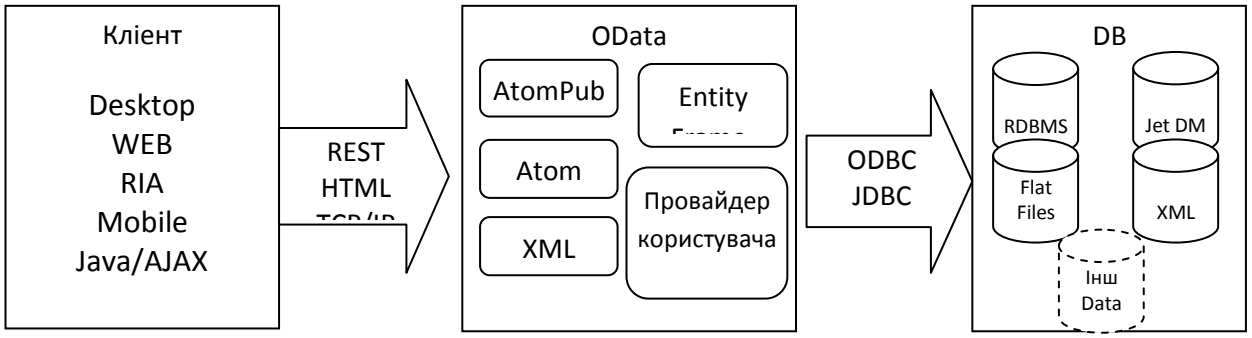

Рис. 1. Загальна схема протоколу OData

Як видно з рис. 1 OData забезпечує високу інтероперабельність – будь-який клієнт може отримати доступ до даних з будь-якого сховища. OData працює таким чином. Він складається з 4 основних компонентів:

модель OData даних, яка надає універсальний спосіб організації та опису даних;

OData протокол, який дозволяє клієнту робити запити та отримувати відповіді від служби OData. По суті, OData протокол являє собою набір базових операцій RESTful на основі HTTP. Базові операції включають звичайні створення (Create), читання (Read), оновлення (Update), видалення (Delete) CRUD) операцій, та OData розширення (мова запитів). Дані, одержувані від служби OData може бути представлена в XML, Atom / AtomPub ,JSON;

клієнтський код доступу до даних;

сервіс OData, який описує кінцеву точку доступу та дозволяє отримати доступ до даних. Сервіс реалізується OData протокол, а також використовує абстрактну модель OData даних для взаємодії з джерелом даних, яке може бути СУБД, списків SharePoint і.т.д;

Загальна схема представлена на рис. 2.

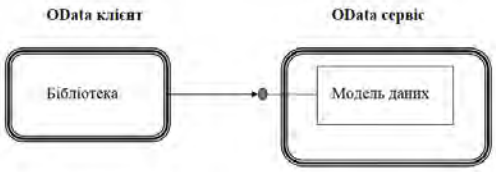

Рис .2. Загальна схема роботи протоколу OData

Якщо розглянути модель даних OData (OData Data Model) глибше,то для забезпечення будь-якому клієнту доступу до будь-якої інформації, OData надає абстрактну модель даних. Причому, дані поставляється в різних формах, і вона може бути пов'язана з іншими даними в різних формах. Для цього OData використовує Entity Data Model, тобто модель сутність-зв'язок, в якій окрім даних, згрупованими за сутностями, зберігаються асоціативні зв’язки між ними. Таким чином, OData працює майже з будь-яким видом даних.

EDM описує тільки логічну структуру даних. Як дані зберігаються фізично не має значення. Дані, надані сервісом OData можуть надходити з багатьох джерел, і як ці дані відображаються у EDM модель, залежить тільки від розробника сервісу. Наприклад, служби OData яка надає реляційні дані може представляти кожну таблицю як єдине ціле, де зовнішній ключ таблиці буде виступати відношення між цими таблицями, що моделюються як асоціації.

Кожен об'єкт є частиною набору сутностей, і кожен набір сутностей належить до певного контейнера сутностей. Структура сутності контейнера проста: дані, що описують сутність та властивості навігації. Властивості навігації представляють асоціацій - вони здійснюють зв'язок між сутностями як з одного контейнеру так і з різних. Наприклад, контейнери сутностей можуть бути таблиці в реляційній базі даних, де кожний рядок представляє окрему сутність у цій таблиці. Навігаційні властивості представляють відносини між рядками (сутностями), у даному випадку вони будуть зовнішніми ключами.

Загальна модель для всіх видів даних, має важливе значення. Без неї OData не може дати клієнтам загальний представлення, єдиний інтерфейс доступу до різноманітних даних. Така концепія є великою

перевагою, коли потрібно працювати з великим об'ємами даних з різних джерел даних. Оскільки тепер можна зв'язати дані з різних серверів в єдине зв'язне ціле, не зважаючи на їх тип , формат, тип сховища. Клієнту не потрібно підключатися до різних сервісів, які використовують різні типи сховищ даних, та збирати дані як мозаїку. Він має єдину точку , в якій вже описана модель. Однією з переваг OData є підтримка усіх сучасних форматів даних: XML, JSON, AtomPub. Причому OData робить їх усіх розширюваними за рахунок додаткових метаданих. Переваг у використанні якогось з перерахованих форматів немає. У протоколі OData вони всі несуть однакове смислове навантаження. Але що стосується формату JSON, то він позбавився своєї простоти в зв'язку з додатковими метаданими. OData версії 3 підтримує Light JSON (JSON у звичайному вигляді).

Можна зробити висновок, що даний протокол забезпечує побудову єдиної абстрактної моделі з багатьох джерел даних різних форматів , модель описується на сервері, а з клієнтського додатку просто іде доступ до неї через RESTful сервіс. Тобто тепер не потрібно описувати модель на стороні клієнта для кожного джерела даних, не потрібно писати свої абстракції. Модель одна як для сервера так і для клієнта, описується один раз і зберігається на сервері. В OData закладена ідея міні-ODBC, але спеціально призначеного для Web. Вчені використали стару ідею у новому середовищі. На початку 1990 років існувало декілька постачальників баз даних, кожен з яких мав власний інтерфейс. Якщо додатку було необхідно підключатися до кількох джерел даних, для взаємодії з кожною з баз даних був необхідний нестандартний код. Для вирішення цієї проблеми Microsoft та ряд інших компаній створили стандартний інтерфейс для отримання і відправки даних джерелам даних різних типів. Цей інтерфейс отримав назву *open database connectivity* (відкритий зв'язок з базами даних). За допомогою ODBC програмісти могли розробляти застосунки для використання одного інтерфейсу доступу до даних, не турбуючись про тонкощі взаємодії з кількома джерелами. Це дозволяє виконати ремапинг даних, адаптувати дані для представлення клієнту, накласти необхідні обмеження для користувачів.

Висновки

OData дозволяє клієнтам побудувати URI, що містить набір сутностей та взаємозв'язки між ними, фільтри. Тобто протокол дозволяє сервісу підтримувати CRUD (Create, Update, Delete operations). Наприклад, для десктопного додатку , щоб отримати доступ до даних з БД використовували SQL запити, в OData теж саме, але не SQL запити а URI з додатковими розширеннями до веб – сервісу. ODBC дозволяє клієнтам ініціювати операції по декількох запитах. Стандартний REST підхід не дозволяє цього, оскільки це порушило його обмеження. Стандартний REST не потребує цього, тому що він надає оброблену інформацію. Тобто якщо додаток підтримуєте дані про погоду, REST надає вам простий спосіб отримати "Погода сьогодні", "погода минулого тижня в Детройті", "Середній рівень опадів в Орландо на червень місяць". Тобто стандартний REST надає оброблену інформацію, ODBC – доступ до усієї інформації. У додатку ODBC клієнт робить що-небудь розумне з даними, перш ніж представити його користувачеві. У REST додатків, як правило, клієнт «презентабельно» відображає вже оброблену інформацію. REST не зовсім підходить для підтримки CRUD. Тому, якщо сервіс надає оброблену інформацію, то OData протокол є збитковим. Але якщо потрібно організувати CRUD, то OData стає корисним. Використання ремапингу моделі даних для Веб дозволяє вирішувати дуже важливі питання з розрізненістю сховищ баз даних, використовуючи механізми ремапингу моделі даних. Гнучкість протоколу OData дозволяє використовувати ці механізми каскадно, тим самим розділяти доступ на бази даних та користувачів сервісів.

Список літератури

1. Introducing OData [Електронний ресурс]. – MSDN. – Режим доступу до ресурсу: http://msdn.microsoft.com/en-us/data/hh237663.aspx.

2. Open Data Protocol [Електронний ресурс]. – Open Data Portal. – Режим доступу: http://www.odata.org.

3. Open Data Protocol [Електронний ресурс] // BizCoder. – Режим доступу до ресурсу: http://www.bizcoder.com/index.php/2009/11/30/oh-data.

4. ODBC [Электроний ресурс] // Wikipedia. – Режим доступу: http://en.wikipedia.org/wiki/ODBC.

5. Open Data Protocol [Електронний ресурс] // WCF Data Service Blog. – Режим доступу до ресурсу: http://blogs.msdn.com/b/astoriateam/archive/2009/11/17/breaking-down-data-silos-the-open-data-protocol-odata.aspx.

Надійшла до редколегії 1.10.2012

Рецензент: д-р екон. наук, проф. А.А. Пилипенко, Харківський національний економічний університет, Харків.

ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ ПРОТОКОЛА ODATA В РАСПРЕДЕЛЕННЫХ ПРИЛОЖЕНИЯХ С ИСПОЛЬЗОВАНИЕМ МОБИЛЬНЫХ УСТРОЙСТВ

А.А. Поляков, В.М. Федорченко, А.П. Ищенко

Современные подход использования мобильных устройств в корпоративной инфраструктуре требует доступа к большому объему данных, но небольшие размеры памяти на устройстве, актуальность и конфиденциальность данных требуют централизованного хранения. Существование различных типов баз данных к которым необходим доступ требует промежуточного слоя отображения модели данных. Механизмы протокола доступа к данным OData позволяет осуществлять ремапинг данных, а также предоставлять избирательной доступ к данным.

Ключевые слова: *распределенная обработка данных, REST, ремапинг данных, протокол OData.*

USING ODATA PROTOCOL IN DISTRIBUTED SYSTEMS WITH MOBILE DEVICES AS A CLIENT LAYER

A.O. Polyakov, V.M. Fedorchenko, A.P. Ischenko

The modern approach is to use mobile devices in the enterprise infrastructure. But such client requires access to large amounts of data, but the have small size of the memory on the device. Also the topicality and confidentiality of data require centralized storage of it. Connection to different types of database with data you want to get requires a middleware data model mapping. Odata protocol provides mechanisms for data access allows remaping of data, and provide selective access to data.

Keywords: *distributed data processing, REST, data remapping, OData protocol.*