Алгоритм

У даній лабораторній роботі як засіб управління повідомленнями розглядається програмний продукт WebSphere MQ компанії IBM. На сьогоднішній день WebSphere MQ є одним з лідерів світового IT-ринку в області технологій MOM. WebSphere MQ – це система черг повідомлень, що дозволяє будувати надійні розподілені системи на основі різнорідних додатків. WebSphere MQ може виконуватися на великій кількості платформ (z/OS, OS/390, MVS, OS/400, Sun Solaris, Linux, AIX, HP-UX, VSE/ESA, OS/2, Windows 95/98, Windows NT/2000/XP і ін.) і підтримує більшість стандартних протоколів зв’язку (в тому числі TCP/IP, NetBIOS, LU62 і SPX). WebSphere MQ забезпечує зберігання і транспортування повідомлень, гарантуючи при цьому доставку критично важливої ​​інформації. Серед основних можливостей даної системи слід виділити наявність механізмів транзакційної обробки даних (WebSphere MQ може виступати в якості менеджера розподілених транзакцій), підтримку стандартних угод захисту каналів даних (в т.ч. SSL), а також наявність програмних інтерфейсів і бібліотек для більшості сучасних мов і середовищ програмування.

Менеджер черг – головний серверний компонент WebSphere MQ – служба, що керує чергами повідомлень і обробкою повідомлень і приймає виклики від прикладних програм. Менеджер черг також іноді називають адміністратором черг. В рамках однієї системи черг повідомлень може бути створено кілька менеджерів черг. У кожного менеджера є своє унікальне ім’я. Черги створюються в рамках менеджера черг. Застосунок, з яким потрібно звернутися до черги для запису або читання повідомлень, має попередньо встановити з’єднання з відповідним менеджером черг WebSphere MQ. Підтримку віддалених з’єднань з менеджером черг забезпечує спеціальна служба Listener ( «одержувач запитів»). Listener виконується для заданого протоколу зв’язку і визначає параметри з’єднання з цього протоколу. Для протоколу TCP/IP Listener за замовчуванням прослуховує порт 1414.

Як вже говорилося вище, черги призначені для проміжного зберігання повідомлень. Фізично черга може бути представлена ​​різними способами: черга може зберігатися в оперативній пам’яті комп’ютера або на диску. У будь-якому випадку, управління чергами виконується менеджером черг і приховане від застосунків. Кожна черга характеризується назвою, і має набір атрибутів, що відповідають за зберігання і обробку повідомлень. Застосунки отримують доступ до черг за допомогою програмних інтерфейсів, в основі яких лежать виклики MQI1, і виконують такі операції з чергою:

1. відкривати чергу для запису або читання повідомлень;
2. поміщати повідомлення в чергу;
3. отримувати повідомлення з черги;
4. переглядати повідомлення, що знаходяться в черзі, без їх вилучення;
5. запитувати і встановлювати атрибути черзі;
6. закривати чергу.

Для того щоб помістити повідомлення в чергу або витягти його з черги, застосунок повинен попередньо відкрити цю чергу на запис або на читання повідомлень, а після завершення роботи з чергою – закрити її. WebSphere MQ підтримує різні типи черг:

* 1. локальні черги – забезпечують зберігання повідомлень;
  2. віддалені черги - використовуються для передачі повідомлень на інші менеджери черг;
  3. псевдочерги – виступають в якості альтернативного імені для іншої черги;
  4. модельні черги - використовуються в якості шаблону при створенні інших черг.

При виконанні лабораторної роботи були використані локальні черги.

Канал – логічна лінія зв’язку, що з’єднує віддалений застосунок WebSphere MQ з менеджером черг, а також два менеджера черг між собою. Канали діляться на дві категорії:

* 1. Канали MQI;
  2. Канали повідомлень.

Канал MQI забезпечує взаємодію між клієнтом WebSphere MQ і менеджером черг. Він призначений для передачі викликів MQI від клієнтського застосунка менеджеру черг і відповідей на ці виклики від менеджера черг клієнту. Іншими словами, канал MQI використовується застосунком при приміщенні і витяганні повідомлень з черги. Канал повідомлень використовується для передачі повідомлень між двома менеджерами черг. При цьому канал повідомлень, на відміну від каналу MQI, є односпрямованим, тобто дозволяє передавати повідомлення в строго певному напрямку.

Після встановлення WebSphere MQ для створення нового менеджера черг було використано утиліту WebSphere MQ Explorer (Оглядач WebSphere MQ). WebSphere MQ Explorer – це графічний засіб адміністрування WebSphere MQ, розроблений на базі платформи Eclipse. Дана утиліта встановилася на комп’ютер разом з продуктом WebSphere MQ. Список менеджерів черг був представлений в утиліті у вікні Навігатор в папці Адміністратори черг. Для створення нового менеджера черг було викликано контекстне меню в цій папці і викликано команду Створити. У майстері, що з’явився було вказано назву менеджера черг. Для інших параметрів значення були залишені за замовчуванням. Щоб створити нову локальну чергу, в утиліті WebSphere MQ Explorer було вибрано у дереві Навігатора всередині потрібного менеджера черг папку Черги і викликано для неї контекстне меню. У меню було вибрано команду Створити → Локальна чергу. На екрані з’явилося діалогове вікно створення черги. Було вказано назву черги і натиснуто кнопку Готово.

В цій лабораторній роботі був використаний інтерфейс Base Java MQ. Класи Base Java MQ містяться в JAR-файлах com.ibm.mq.jar і com.ibm.mq.jmqi.jar, що розміщуються в директорії \Program Files\IBM\WebSphere MQ\java\lib\. Дані JAR-файли були підключені до мого проекту Java, щоб працювати з системою WebSphere MQ.

Бібліотека com.ibm.mq містить наступні основні класи, що забезпечують роботу з системою WebSphere MQ:

* 1. MQEnvironment – використовується для налаштування параметрів з’єднання з менеджером черг;
  2. MQQueueManager – клас «менеджер черг» – дозволяє встановити з’єднання з менеджером черг і звертатися до черг;
  3. MQQueue – клас «черги» – дозволяє відправляти і приймати повідомлення;
  4. MQMessage – клас «повідомлення» – дозволяє формувати і зчитувати вміст прикладної частини і заголовка повідомлення;
  5. MQGetMessageOptions – клас «опції читання» – задає додаткові параметри для операції читання повідомлення з черги;
  6. MQPutMessageOptions – клас «опції запису» – задає додаткові параметри для операції запису повідомлення в чергу;
  7. MQMD – клас, що описує заголовок повідомлення;
  8. MQC – містить константи WebSphere MQ;
  9. MQException – виняток WebSphere MQ.

Для налаштування параметрів з’єднання з менеджером черг використовувалися статичні поля класу MQEnvironment:

* 1. hostname – ім’я хоста або IP-адреса;
  2. port – номер порту TCP/IP;
  3. channel – назва каналу MQI;
  4. CCSID – кодова сторінка, яка використовувалася для представлення даних.

Для взаємодії з менеджером черг по каналу MQI (при підключенні до віддаленого менеджеру черг) було задано ненульове значення для параметра hostname, а також вказано назву каналу MQI в параметрі channel (використовувався канал за замовчуванням з назвою SYSTEM.DEF.SVRCONN).

Для обробки помилок, що виникали при роботі з WebSphere MQ, використовувався клас MQException. Клас MQException успадкований від класу java.lang.Exception і використовується в конструкції try – catch. Клас має два основних поля:

* completionCode – код завершення операції
* reasonCode – код помилки

Поле completionCode приймало значення:

* MQException.MQCC\_WARNING – операція виконалася з попередженнями
* MQException.MQCC\_FAILED – операція завершилася помилкою

Поле reasonCode використовувалося для уточнення причини виникнення помилки. Клас також містить метод getMessage(), який повертає повідомлення про помилку.

З’єднанням з менеджером черг управляв клас MQQueueManager. Для установки з’єднання з менеджером черг було створено об’єкт класу MQQueueManager. При цьому в конструктор класу передавалася назва менеджера черг. Параметри з’єднання визначалися при створенні екземпляра класу MQQueueManager на основі значень полів класу MQEnvironment. Для розриву з’єднання використовувався метод класу disconnect().

Щоб відправляти і читати повідомлення, необхідно було попередньо відкрити потрібну чергу. Для відкриття черги використовувався метод accessQueue екземпляра класу MQQueueManager. У метод accessQueue передавалася назва черги queueName і опції відкриття черги openOptions. Для визначення опцій відкриття черги використовувалися константи MQOO\_\* класу MQC:

* MQC.MQOO\_OUTPUT – відкриття черги для запису повідомлень;
* MQC.MQOO\_BROWSE – відкриття черги для перегляду повідомлень;
* MQC.MQOO\_INPUT\_EXCLUSIVE – відкриття черги для отримання повідомлень (ексклюзивний режим, тільки один процес може читати повідомлення з черги);
* MQC.MQOO\_INPUT\_SHARED – витяг повідомлень (режим загального доступу);
* MQC.MQOO\_INPUT\_AS\_Q\_DEF – витяг повідомлень (стандартний режим).

На основі представлених констант необхідно було сформувати цілочисельне значення openOptions, використовуючи побітову логічну операцію «або». Наприклад, якщо ми хочемо відкрити чергу для читання і перегляду повідомлень, як параметр openOptions можна вказати таке значення: «MQC.MQOO\_INPUT\_EXCLUSIVE | MQC.MQOO\_BROWSE». У випадку успішного виконання метод accessQueue повертає екземпляр класу MQQueue. По завершенні роботи з чергою її слід закрити за допомогою методу close.

Для роботи з повідомленнями WebSphere MQ використовувався клас MQMessage. Даний клас дозволяє формувати прикладну частину повідомлення і працювати з його заголовком. Клас MQMessage використовувався при записі повідомлень в чергу і при читанні повідомлень з черги. Робота з прикладною частиною повідомлення здійснювалася через механізм потокового введення-виведення за допомогою методів виду readXXX (читання даних з прикладної частини) і writeXXX (запис даних в прикладну частину). Ці методи визначені в стандартних Java-інтерфейсах DataInput і DataOutput. Параметри заголовка повідомлення описувалися класом MQMD. Клас MQMessage успадкований від класу MQMD і містить поля, що дозволяють працювати з заголовком повідомлення, зокрема:

* messageID – унікальний ідентифікатор повідомлення (byte[]);
* correlationID – ідентифікатор зв’язку, дозволяє об’єднати кілька повідомлень в групу (byte[]);
* format – формат прикладної частини повідомлення (String);
* messageType – тип повідомлення (int);
* persistence – «сталість» повідомлення (int) – визначає, чи буде повідомлення збережено на жорсткий диск під час запису в чергу, або воно буде зберігатися в оперативній пам’яті;
* priority (int) – пріоритет повідомлення.

Щоб створити в чергу використовувався метод put класу MQQueue. У метод передавалося посилання на повідомлення MQMessage, яке необхідно помістити в чергу. У WebSphere MQ існує можливість відправляти повідомлення без явного відкриття черги безпосередньо через об’єкт менеджера черг. У класу MQQueueManager є метод put, який фактично виконує три операції: відкриває чергу, записує в чергу повідомлення і закриває чергу. У метод передавалося посилання на повідомлення MQMessage і назва черги, в яку це повідомлення було потрібно записати. Метод put менеджера черг я використовував в тому випадку, коли потрібно було записати єдине повідомлення в чергу. Якщо було потрібно записувати в процесі роботи програми кілька повідомлень в одну і ту ж чергу, даний спосіб запису був неефективний, так як кожен раз при його виклику черга знову відкривалася і закривалася, що потребувало додаткового часу.

Щоб прочитати повідомлення з черги використовувався метод get класу MQQueue. Метод витягував повідомлення з черги і записував його в заданий об’єкт MQMessage. У тому випадку, якщо черга порожня, метод генерував виняток MQException з кодом причини MQException.MQRC\_NO\_MSG\_AVAILABLE (код 2033). При читанні повідомлення з черги вказувались додаткові опції, використовуючи клас MQGetMessageOptions. Даний клас дозволяє:

* визначити інтервал очікування повідомлення в черзі;
* встановити використання транзакцій при читанні повідомлення;
* визначити критерії пошуку повідомлення в черзі;
* встановити режим перегляду повідомлень (повідомлення будуть прочитані без видалення з черги).

Поле waitInterval класу MQGetMessageOptions задавало інтервал очікування надходження повідомлення в чергу і використовувалося разом з опцією MQC.MQGMO\_WAIT. В даному випадку, метод get при наявності повідомлення в черзі відразу ж повертав його програмі. Якщо чергу виявлялася порожньою, метод протягом встановленої кількості секунд чекав надходження повідомлення, і якщо повідомлення за цей час не приходило, метод генерував виняток MQException. Для параметра waitInterval встановлювалося спеціальне значення MQC.MQWI\_UNLIMITED. В такому випадку метод get чекав надходження повідомлення необмежений час.

Взаємодія клієнта і сервера відбувалася за наступним сценарієм:

1. Застосунок-клієнт запускався в перший раз (метод Client.main1) і відправляв серверу кілька запитів на виконання запитів з бази даних Map, після чого завершував свою роботу.
2. Запускався сервер. Він отримував запити клієнта, проводив необхідні витяги із бази даних Map, формував і відправляв відповіді, і завершував свою роботу.
3. Клієнт запускався знову (метод Client.main2) і отримував результати відправлених раніше запитів від сервера.