

Wielowątkowość i JMS

Warsztaty - dzień 5

v3.0

Plan

1. Warsztaty - wielowątkowość i JMS
2. Przygotowania
3. Aplikacja - problem uczących filozofów
4. Aplikacja - wielowątkowy odczyt i zapis danych
5. Aplikacja JMS - składanie zamówień
6. Aplikacja JMS - kanał informacyjny
7. System kolejkowy - aplikacja webowa

Warsztaty - wielowątkowość i JMS

Celem warsztatów jest napisanie czterech funkcjonalnych aplikacji rozwiązujących konkretne problemy. Aplikacje, które zostaną utworzone podczas warsztatów:

- rozwiązanie problemu uczących filozofów,
- aplikacja jednocześnie odczytująca i zapisująca dane do i z plików txt,
- aplikacja JMS - składanie zamówień,
- aplikacja JMS - kanał informacyjny.

Przygotowania

Przygotowania

Przed przystąpieniem do każdego z zadań przygotuj:

- osobny projekt dla każdej aplikacji
- uzupełnij zestaw zależności odpowiedzialnych za Spring MVC
- uzupełnij podstawowy zestaw zależności dla korzystania z Hibernate i Spring Data
- załóż nowe repozytoria Git na GitHubie i nowe bazy danych (jeżeli są potrzebne)
- stwórz plik .gitignore i dodaj do niego elementy ignorowane np. podstawowe pliki projektu (.project, .settings)
- możesz skorzystać z serwisu <https://www.gitignore.io/>

Aplikacje do stworzenia

Aplikacja - problem ucztujących filozofów

Aplikacja - problem ucztujących filozofów

Aplikacja ta będzie wykonywana w konsoli

Problem ucztujących filozofów (znany też jako problem pięciu filozofów) jest klasycznym przykładem używanym do przedstawienia problemów synchronizacji danych w wielowątkowych środowiskach

Problem polega na tym że przy stoliku siedzi pięciu filozofów(dla ułatwienia można ich oznaczyć symbolami od P1 do P5) i jedyne co robią to jedzą i myślą. Przed filozofami leży 5 widelców i aby mieć możliwość jedzenia filozof musi mieć dwa widelce w rękach.

Aplikacja - problem ucztujących filozofów

Po zjedzeniu posiłku przez danego filozofa, odkłada on obydwa widelce i mogą one zostać podniesione przez innego filozofa (pomijamy kwestie mycia sztućców) który powtarza tą samą czynność

Problem polega na tym aby wykonać taki algorytm który pozwoli każdemu filozofowi zjeść posiłek i nie pozwolić im czekać na zjedzenie posiłku w nieskończoność (np. deadlock)

Aplikacja - problem ucztujących filozofów

Jako schemat działania w aplikacji wykorzystaj poniższy pseudo kod:

```
while(true) {  
    // filozof myśli  
    think();  
    // proces jedzenia  
    pick_up_left_fork();  
    pick_up_right_fork();  
    eat();  
    put_down_right_fork();  
    put_down_left_fork();  
    // po zakończeniu jedzenia powrót do myślenia  
}
```

Aplikacja - problem uczących filozofów

W implementacji problemu filozofów można użyć przykładowej konstrukcji klasy implementującej interfejs Runnable

```
public class Philosopher implements Runnable {  
    private Object leftFork;  
    private Object rightFork;  
    public Philosopher(Object leftFork, Object rightFork) {  
        this.leftFork = leftFork;  
        this.rightFork = rightFork;  
    }  
    @Override  
    public void run() {  
        // TODO  
    }  
}
```

Aplikacja - problem ucztujących filozofów

W powyższej klasie należałoby również zawrzeć metodę/metody która informuje o wykonywanej akcji, można to zrobić za pomocą trzech różnych metod lub jednej uproszczonej:

```
private void doAction(String action) throws InterruptedException {  
    System.out.println(  
        Thread.currentThread().getName() + " " + action);  
    Thread.sleep(((int) (Math.random() * 100)));  
}
```

Aplikacja - problem ucztujących filozofów

W metodzie która będzie uruchamiała przykład tworząc obiekty filozofów(Philosopher) oraz sztuców(Object) pamiętaj o poprawnym przyporządkowaniu poszczególnych sztuców do konkretnych filozofów, użyj do tego pętli.

Aplikacja - wielowątkowy odczyt i zapis danych

Aplikacja - wielowątkowy odczyt i zapis danych

Daną wejściową do aplikacji będzie nazwa pliku tekstowego zawierającego pewne dane, taki plik należy stworzyć samemu.

Aplikacja ta będzie wykonywała się w konsoli i jej zadaniem będzie odczytanie danych z podanego w parametrze pliku za pomocą jednego wątku.

W tym samym czasie drugi wątek będzie korzystał z odczytanych danych przez pierwszy wątek i pobierając te dane będzie je zapisywał do drugiego pliku tekstowego o innej nazwie od pierwszego

Aplikacja - wielowątkowy odczyt i zapis danych

Przy wykonywaniu zadania podziel je na dwie części:

- klasa obsługująca odczytanie pliku
- klasa obsługująca zapis do nowego pliku

Klasy te powinny być połączone wspólnym elementem którym może być obiekt **BlockingQueue**

Aplikacja - wielowątkowy odczyt i zapis danych

Najbardziej przydatne obiekty do wczytania pliku to np:

- BufferedReader
- FileReader
- File

Natomiast przy zapisie danych do pliku wykorzystaj obiekty takie jak:

- PrintWriter
- File

Aplikacja - wielowątkowy odczyt i zapis danych

Nie zapomnij o obsłudze typowych błędów:

- FileNotFoundException
- IOException
- InterruptedException

Aplikacja JMS - składanie zamówień

Aplikacja JMS - składanie zamówień

Celem stworzenia aplikacji do składania zamówień wykorzystującej JMS będzie przedstawienie praktycznego przykładu komunikacji Point-To-Point

Aplikacja będzie wykorzystywała Spring MVC oraz Hibernate i Spring Data

W aplikacji będą znajdowały się trzy widoki:

- widok klienta składającego zamówienie
- lista zamówień
- widok do przyjmowania zamówień

Aplikacja JMS - składanie zamówień

Aplikacja będzie wykorzystywała tabelę produktów oraz tabelę zamówień

Tabela będzie zawierała:

- id - klucz główny
- id_klienta - pole typu integer
- id_produktu - klucz obcy do tabeli produktów
- potwierdzenie - pole typu boolean

Tabela Produkty będzie zawierała:

- id - klucz główny
- nazwa - nazwa produktu

Aplikacja JMS - składanie zamówień

Utwórz kontroler, gdzie w widoku Klienta będzie znajdowała się lista wyboru produktów wygenerowana z tabeli Produkty

Po wybraniu produktu z listy użytkownik kliknie przycisk "Zamów"

Wtedy zostanie dodany nowy rekord do tabeli Zamówienia z id Klienta pobranym z paska adresu oraz id produktu pobranego z listy wyboru, pole potwierdzenie będzie puste co trzeba uwzględnić w interfejsie użytkownika i wyświetlić komunikat "nie potwierdzone"

Następnie użytkownik zostanie przeniesiony do strony z listą zamówień dla danego Klienta

W tym samym czasie za pomocą JMS zostanie wysłana wiadomość do odpowiedniej kolejki zbierającej zamówienia

Aplikacja JMS - składanie zamówień

Utwórz również kontroler dla widoku przyjmowania zamówień

Za każdym razem po odświeżeniu tego kontrolera w tle będą pobierane wiadomości z kolejki, a następnie dla każdego nowego zamówienia będzie aktualizowana kolumna potwierdzająca zamówienie produktu

Zadbaj o dodanie ewentualnych dodatkowych parametrów przesyłanych za pomocą JMS

Aplikacja JMS - składanie zamówień

Po potwierdzeniu przyjęcia zlecenia wykonanym w poprzednim slajdzie, Klient po ponownym wejściu na stronę "lista zamówień" zobaczy listę swoich zamówień, tym razem kolumna potwierdzenie będzie wypełniona

Należy to uwzględnić w interfejsie użytkownika i wyświetlić tę informację w przyjazny dla użytkownika sposób, dodając przy okazji komunikat "Zamówienie potwierdzone"

Aplikacja JMS - kanał informacyjny

Aplikacja JMS - kanał informacyjny

Celem wykonania aplikacji kanał informacyjny będzie praktyczne pokazanie zastosowania JSM w modelu Publish/Subscribe

Aplikacja będzie zawierała dwa widoki:

- widok wysyłania wiadomości do subskrybentów
- widok odczytu wiadomości przez subskrybenta

Aplikacja JMS - kanał informacyjny

Aplikacja będzie zawierała dwie tabele:

Tabela Subskrybenci:

- id - klucz główny,
- temat - pole typu string zawierające temat, na który jest wykonana subskrypcja,
- uzytkownik - nazwa użytkownika, który jest zapisany na dany temat,

Tabela Subskrypcje:

- id - klucz główny,
- tresc - pole typu string zawierające treść przesłanej wiadomości,
- uzytkownik - nazwa użytkownika do którego dana wiadomość była przypisana,
- timestamp.

Aplikacja JMS - kanał informacyjny

Po wejściu na widok wysyłania wiadomości do subskrybentów aplikacja będzie w interfejsie użytkownika zawierała pole do wpisania wiadomości oraz przycisk do wysłania wiadomości.

Po wysłaniu wiadomości aplikacja będzie łączyła się z odpowiednim tematem i wysyłała do niego wiadomość, jednocześnie każdy z subskrybentów będzie w tle odbierał wiadomość.

Odebrana przez subskrybenta wiadomość będzie zapisywana w bazie danych w tabeli Subskrypcje.

Aplikacja JMS - kanał informacyjny

Po wejściu na widok odczytu wiadomości przez subskrybenta prezentowana będzie lista wiadomości które użytkownik otrzymał na subskrybowany temat

Lista wiadomości będzie pobierana z tabeli Subskrypcje po polu użytkownik który w kontrolerze będzie podawany w pasku adresu

Lista wiadomości powinna być wyświetlona w formie tabeli i sortowana od najnowszych do najstarszych wiadomości

System kolejkowy - aplikacja webowa

System kolejkowy

Celem aplikacji jest wykonanie systemu kolejkowego służącego do wgrywania plików, zipowania ich i późniejszego pobierania.

Po wejściu na stronę główną użytkownik ma zobaczyć formularz i za jego pomocą ma mieć możliwość wybrania kilku plików, dodania nazwy wynikowego pliku zip i naciśnięcia przycisku np. "Zip all files"

Po naciśnięciu wcześniej wspomnianego przycisku dzieją się dwie rzeczy:

- Użytkownik zostaje przeniesiony na stronę na której zostaje wygenerowany dla niego numer zwany dalej numerem kolejkowym.
- W tle pliki przesłane przez użytkownika zostają zipowane w jeden plik o nazwie podanej przez użytkownika w formularzu na stronie głównej.

System kolejkowy

Użytkownik ma możliwość przejścia na stronę pobierania plików na której znajduje się: pole tekstowe do wpisania wcześniej otrzymanego numeru kolejkowego oraz przycisk np. "Check zip status"

Po wprowadzeniu numeru i kliknięciu przycisku, system sprawdza czy operacja zip-owania wcześniej przesłanych plików zakończyła się.

W przypadku zakończenia zip-owania plików użytkownik przenoszony jest na stronę z linkiem do pobrania wygenerowanego archiwum zip, w przeciwnym wypadku otrzymuje komunikat o braku możliwości pobrania pliku z prośbą o późniejsze sprawdzenie statusu operacji zip-owania

System kolejkowy

Aby łatwiej było Ci napisać aplikację rozbij jej tworzenie na kilka etapów.

Etap pierwszy - opracuj mechanizm zipowania. Do jego utworzenia wykorzystaj takie klasy jak np:

- File
- FileOutputStream
- FileInputStream
- ZipOutputStream
- ZipEntry

Obsłuż również wyjątki **IOExceptions**

System kolejkowy

Etap drugi - zbuduj mechanizm kolejkowy w którym zapewnij generowanie kolejnych unikalnych numerów kolejkowych.

Ważne jest również aby dany numer został zapisany w tabeli wraz z nazwą finalnie wygenerowanego pliku zip i linkiem, tak aby można było go później ściągnąć.

Aby zachować niezbędne dane utwórz tabelę o nazwie "pliki_zip" składającą się z pól:

- id - klucz główny tabeli
- nazwa - nazwa wygenerowanego pliku zip
- link - link do ściągnięcia pliku
- nr_kolejki - unikalny numer kolejkowy
- timestamp - data wygenerowania pliku zip

System kolejkowy

Jeżeli przydadzą Ci się inne kolumny w tabeli, nic nie stoi na przeszkodzie aby dodać je do tworzonego schematu

Ten etap powinien również obejmować napisanie mechanizmu odpowiadającego za wyszukanie danego pliku zip na podstawie podanego numeru kolejkowego i wygenerowanie linku do tego pliku

Aby zasymulować długie działanie procesu zipowania możesz zastosować odpowiedni mechanizm opóźnienia w tej metodzie

System kolejkowy

Etap trzeci - wykonaj mechanizm ściągania plików.

Ostatni i teoretycznie najprostszy z etapów, gdzie na podstawie linku generowanego z etapu drugiego ma zostać wykonane ściągnięcie pliku zip.

Wystarczy aby kliknąć link i pobieranie pliku zip powinno rozpocząć się automatycznie. Możesz przekierować użytkownika na stronę z informacją o pobraniu pliku i podziękowaniu za korzystanie z aplikacji

System kolejkowy

Kiedy uda Ci się doprowadzić aplikację do postaci w pełni działającej, postaraj się wprowadzić pewne zmiany udoskonalając jeszcze aplikację np:

- ogranicz wielkość wysyłanych plików
- rozszerz tabelę "pliki_zip" o możliwość przechowywania nazw plików z których dany plik zip został wygenerowany
- zmodyfikuj metodę zip-owania tak aby korzystała ona z mechanizmu wielowątkowości
- dodaj możliwość wylistowania wszystkich wygenerowanych plików zip dla każdego użytkownika korzystającego z aplikacji
- wykorzystaj JMS i użyj np. wysyłania maili do powiadomienia danego użytkownika o zakończeniu procesu zip-owania jego plików

Pomyśl co jeszcze mogłoby przydać się w aplikacji i spróbuj zawrzeć to w swoim projekcie

KONIEC