1.2.4. Використання транзакцій

Транзакція - це серія операцій, які відокремлюються та виконуютсья як атомарне ціле. Операція у рамках транзакції повинна виконатися або повністю, або повернути дані до початкового стану.

Обробка транзакцій призначена для підтримки цілісності системи (як правило, бази даних або деяких сучасних файлових систем) у відомому стані, забезпечуючи, що взаємозалежні операції в системі або успішно завершені, або всі успішно скасовані. Оновна ідея - незалежно від того, що станеться, (вимкнення електроенергії, катаклізм, або збій у роботі програмного забезпечення), дані, з якими ви працюєте, будуть розглядатися атомарно, та змінюватись атомарно.

Наприклад, розгляньте типову банківську транзакцію, яка передбачає переведення 700 у.о. з ощадного рахунку на дебетовий рахунок клієнта. Ця операція передбачає принаймні дві окремі операції в комп'ютерних умовах: зняття з ощадного рахунку 700 у.о. та поповнення поточного рахунку на 700 у.о..

Якщо одна операція успішна, а інша ні, то в кінці дня залишок на картах не будуть співпадати. Отже, існує спосіб забезпечення умов, у яких обидві операції були успішними або обидві невдачі, так щоб в базі даних банку в цілому ніколи не було невідповідності.

Обробка транзакцій пов'язує декілька окремих операцій в єдину, неділиму транзакцію та гарантує, що всі операції в транзакції виконуються без помилок, або жодна з них не виконується. Якщо деякі операції завершені, але виникають помилки, система обробки транзакцій "відкачує" всі операції транзакції (включаючи успішні), тим самим видаляючи всі сліди транзакції та відновлення системи до послідовного початкового стану операції. Якщо всі операції транзакції успішно завершені, операція здійснюється системою, і всі зміни в базі даних стають постійними; транзакцію не можна відкотити після того, коли це буде зроблено.

Обробка транзакцій захищає від апаратних та програмних помилок, які можуть залишити транзакцію частково завершеною. Якщо комп'ютерна система аварійно завершується посередині транзакції, система обробки транзакцій гарантує, що всі операції в будь-яких нездійснених станах скасуються.

2.4. MySQL Connector/J

MySQL забезпечує з'єднання для клієнтських додатків, розроблених на мові програмування Java, за допомогою MySQL Connector / J, драйвера, який реалізує API Java Database Connectivity (JDBC) .

2.5. Хеш-фунції SHA-256

SHA-2 ( Secure Hash Algorithm 2 ) - це набір криптографічних хеш-функцій, розроблений Агентством національної безпеки США (NSA). [3] Вони побудовані з використанням структури Меркле - Дамгарда, з функції одностороннього стиснення, яка сама по собі побудована з використанням структури Девіса-Майєра з (класифікованого) спеціалізованого блочного шифру.

Криптографічні хеш-функції - це математичні операції, що виконуються на цифрових даних; шляхом порівняння обчисленого "хеш" числом з відомим і очікуваним значенням хешу, людина може визначити цілісність даних. Наприклад, обчислення хеш завантаженого файлу та порівняння результату з попередньо опублікованим результатом хешу може показати, файл завантажено чи змінено. Ключовим аспектом криптографічних хеш-функцій є їх стійкість до колізій: ніхто не повинен мати змогу знаходити два різних вхідних значення, що призводять до того ж хеш-вихідного сигналу.

SHA-256 - нова хеш-функція, обчислена за допомогою 32-бітових слів. Усі хеш-функції використовують різні обсяги зсуву та додаткові константи, але їх структури практично ідентичні, відрізняються лише кількість проходів.

SHA-256 являє собою односпрямовану функцію для створення цифрових відбитків фіксованої довжини (256 біт, 32 байт) з вхідних даних розміром до 2,31 ексабайт (2⁶⁴ біт) і є окремим випадком алгоритму з сімейства криптографічних алгоритмів SHA-2 ( Secure Hash Algorithm Version 2 ) опублікованим АНБ США (Агентством національної безпеки США) у 2002 році.

Основна робота цієї хеш-функції полягає в перетворенні (або хешування) набору елементів даних в значення фіксованої довжини. Це значення довжини буде порівнюватися з копіями вихідних даних, без можливості вилучення цих вихідних даних.

Оригінал тексту після доповнення розбивається на блоки, кожен блок - на 16 слів. Алгоритм пропускає кожен блок повідомлення через цикл з 64 ітераціями. На кожній ітерації 2 слова перетворюються, функцію перетворення задають інші слова. Результати обробки кожного блоку складаються, сума є значенням хеш-функції. Так як ініціалізація внутрішнього стану проводиться результатом обробки попереднього блоку, то немає можливості обробляти блоки паралельно.

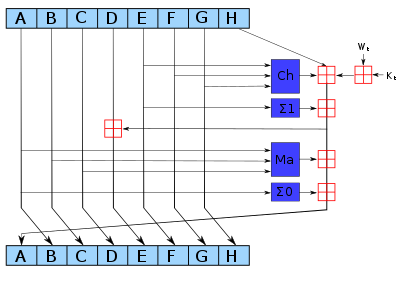


Рис.2.1 Графічне представлення однієї ітерації обробки блоку даних

2.6. Утиліта для ведення журналів Log4j

Apache Log4j - це утиліта для ведення журналів на базі Java. Вона є частиною сервісу протоколювання Apache проекту Apache Software Foundation . Log4j - це одна з декількох систем журналювання Java .

Внесення операторів коду до журналу - це метод низькорівневий метод для його налагодження. Це також може бути єдиним способом, оскільки дебагери не завжди доступні або застосовні. Це, як правило, актуально для багатопоточних додатків та розподілених програм у цілому.

Досвід показує, що ведення журналів є важливим компонентом циклу розробки. Він пропонує кілька переваг. Він забезпечує точний контекст виконання програми. Після вставки в код, генерація виходного журналу не вимагає втручання людини. Більш того, вихідний журнал можна зберегти в постійному середовищі для вивчення пізніше. На додаток до його використання в циклі розробки, достатньо багатий пакет реєстрації також може розглядатися як інструмент аудиту.

Журнал має свої недоліки. Це може уповільнити програму. Якщо це занадто багатослівне протоколювання, це може спричинити незрозумілісь протоколу програми. Щоб полегшити ці проблеми, log4j розроблений таким чином, щоб бути надійним, швидким і розширюваним. Оскільки журналювання рідко є основним завданням програми, API log4j прагне бути простим для розуміння та використання.

2.7. JSTL

JavaServer Pages Standard Tag Library ( JSTL ) є одним з компонентів Java Enterprise Edition веб - додатків платформи розробки. Це розширює

специфікацію JSP , додаючи бібліотеку тегів JSP для звичайних завдань, таких як обробка даних XML , умовне виконання, доступ до бази даних , цикли та інтернаціоналізація .

JSTL є альтернативою такому виду вбудованої в JSP логіки, як скріплет, тобто прямі вставки Java коду. Використання стандартизованого безлічі тегів краще, оскільки одержуваний код легше підтримувати і простіше відокремлювати бізнес-логіку від логіки відображення.

3.1. Java Virtual Machine

Віртуальна машина Java ( JVM ) являє собою віртуальну машину , яка дозволяє комп'ютеру працювати Java - програмам, а також програмам , написаних на інших мовах і інтерпретованих на Java байткод . JVM детально визначається специфікацією, яка формально описує, що вимагається від впровадження JVM. Наявність специфікації забезпечує сумісність програм Java з різними реалізаціями, так що авторам програмне потрібно турбуватися про особливі схеми базової апаратної платформи.

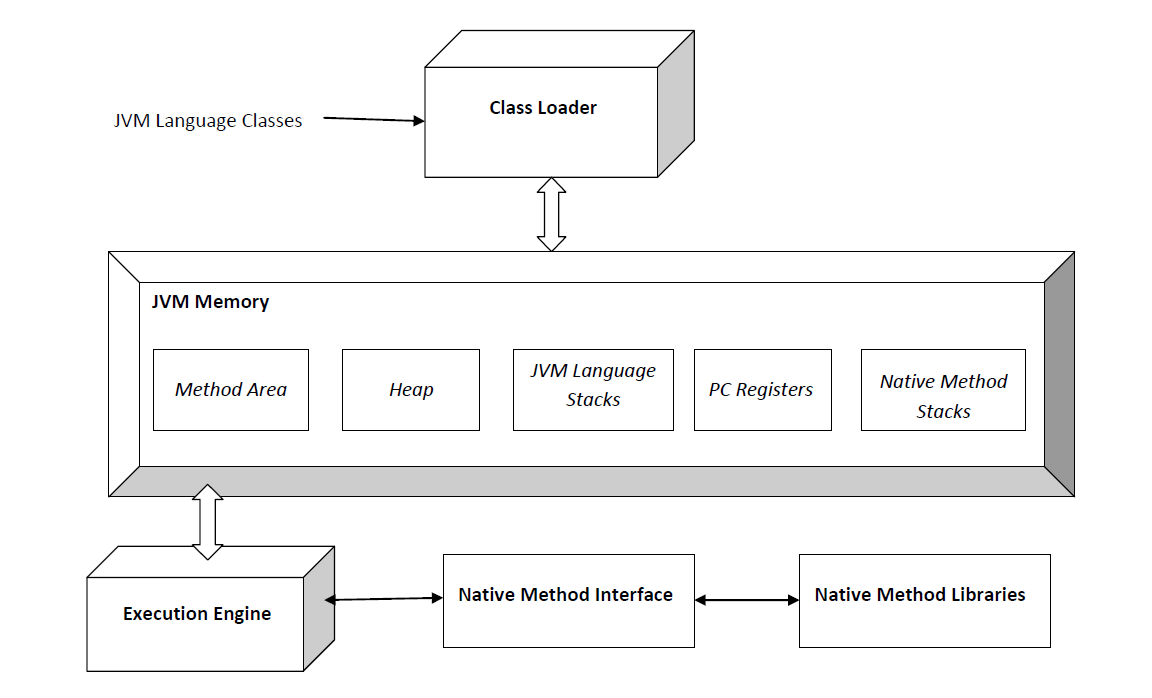


Рис.3.1 Огляд віртуальної машини Java

3.2. Java cервлет

Java сервлет – це Java програма, яка розширює можливості сервера. Хоча сервлети можуть відповідати на будь-які типи запитів, вони найчастіше реалізують програми, розміщені на веб-серверах.

Сервлет Java обробляє або зберігає клас Java, який відповідає стандарту для реалізації класів Java, які відповідають на запити: API Java Servlet. Сервлети можуть в принципі спілкуватися через будь -який протокол мережі, але вони найчастіше використовуються з HTTP протоколом. Таким чином, "сервлет" часто використовується як скорочення "HTTP-сервлет". Таким чином, розробник програмного забезпечення може використовувати сервлет для додавання динамічного вмісту на веб-сервер за допомогою платформи Java . Сформований вміст зазвичай у форматі HTML, але може бути й інші дані, такі як XML, JSON. Сервлет може підтримувати стан необхідних змінних в сесійній змінних сервера з використанням HTTP cookies або URL перезапису .

Для розгортання та запуску сервлета потрібно використовувати веб-контейнер.

3.3 JavaServer Page

JavaServer Page (JSP) - це технологія, яка допомагає розробникам програм створювати динамічно створювані веб-сторінки на основі HTML , XML або інших типів документів.

JSP дозволяє Java-коду та деяким заздалегідь визначеним діям переплітатися з вмістом статичного вмісту веб-розмітки, наприклад, HTML, причому результуюча сторінка збирається та виконується на сервері. Скомпільовані сторінки, а також будь-які залежні бібліотеки Java містять Java-байт-код, а не машинний код.

Як і будь-яка інша програма Java, вона повинна бути виконана в рамках віртуальної машини Java (JVM), яка взаємодіє з операційною системою хосту сервера.

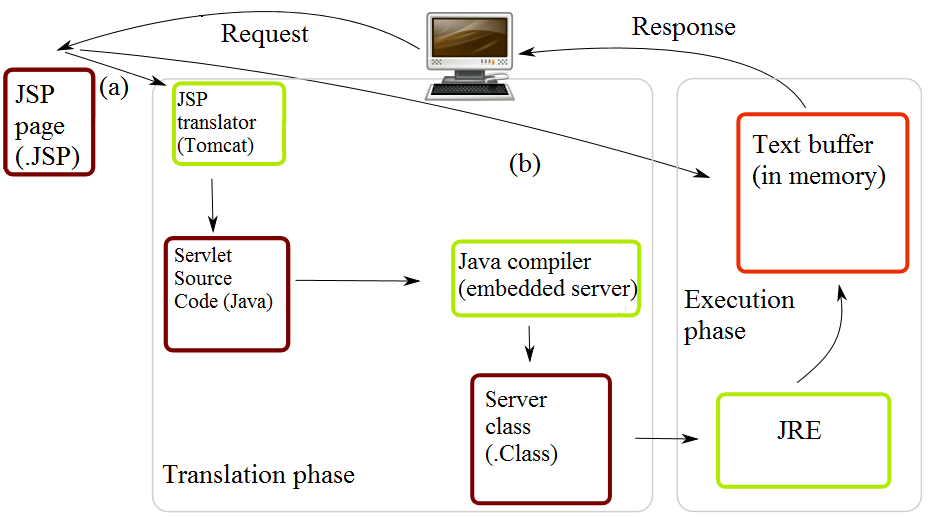


Рис.3.2 Життєвий цикл JSP

3.4. Веб-контейнер

Веб - контейнер (також відомий як контейнер сервлетів) є компонентом веб - сервера , який взаємодіє з Java сервлетами . Веб-контейнер відповідає за управління життєвим циклом сервлетів, відображення URL-адреси до певного сервлета та забезпечення того, що запитувач URL-адреси має правильні права доступу.

Основною ідеєю контейнера Servlet є використання Java для динамічного створення веб-сторінки на стороні сервера. Таким чином, контейнер сервлета по суті є частиною веб-сервера, який взаємодіє з сервлетами.

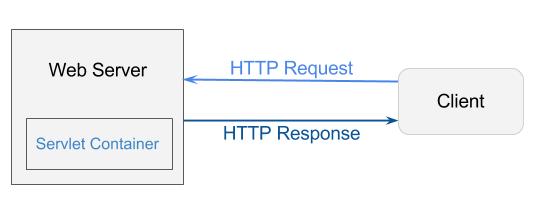


Рис.3.3 Схематичне зображення взаємодії контейнеру сервлетів з веб-сервером

Веб-контейнер обробляє запити до сервлетів, файлів JavaServer Pages (JSP) та інших типів файлів, що містять код сервера. Веб-контейнер створює екземпляри сервлету, завантажує та вивантажує сервлети, створює та управляє об'єктами запиту та відповіді, а також виконує інші завдання керування сервером.

Веб-контейнер реалізує контракт веб-компонентів архітектури Java Enterprise Edition, вказуючи середовище виконання для веб-компонентів, що включає безпеку, паралельність, керування життєвим циклом, транзакцію, розгортання та інші послуги.

Порядок обробки запиту на веб-сервері який використовує контейнер сервлетів:

1. Веб-сервер отримує HTTP-запит
2. Веб-сервер пересилає запит до контейнера сервлета
3. Сервлет динамічно завантажується та завантажується в адресний простір контейнера, якщо він не знаходиться в контейнері.

Контейнер викликає метод init () для ініціалізації сервлета (викликається один раз, коли сервлет завантажується в перший раз)

1. Контейнер викликає метод service () сервлет для обробки запиту HTTP, тобто читає дані у запиті та формує відповідь. Сервлет залишається в адресному просторі контейнера і може обробляти інші HTTP-запити.
2. Веб-сервер повертає динамічно згенеровані результати до потрібного місця

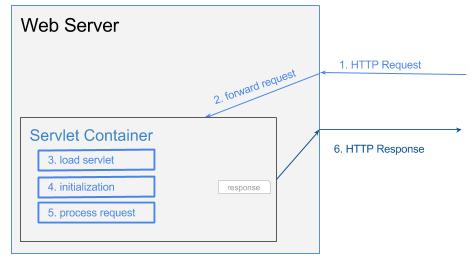


Рис.3.4 шість етапів обробки запиту на веб-сервері

Використання сервлетів дозволяє JVM обробляти кожен запит в окремому потоці Java, і це є однією з основних переваг контейнера сервлетів. Кожен сервлет - це клас Java з особливими елементами, що відповідають HTTP-запитам. Основна функція сервлету полягає в тому, щоб передати запити до сервлета для обробки і повернути динамічно згенеровані результати до правильного розташування після того, як JVM обробить їх. У більшості випадків контейнер сервлета працює з єдиною JVM, але є рішення, коли контейнер потребує декількох JVM.

3.5. Apache Tomcat

Apache Tomcat, часто називається Tomcat Server- це контейнер Java Servlet з відкритим кодом, розроблений Apache Software Foundation (ASF). Tomcat реалізує декілька специфікацій Java Enterprise Edition, включаючи Java Servlet , JavaServer Pages (JSP), Java EL та WebSocket, і забезпечує середовище HTTP веб-сервера можливостями JVM, в якому може працювати код Java.

Tomcat був випущений разом з такими компонентами, як Catalina (контейнер для сервлета), Coyote (HTTP-коннектор) і Jasper ( JSP-двигун ). Проведемо їх аналіз та зїясуємо, які функії вони виконують.

Catalina - контейнер сервлета Tomcat . Каталіна реалізує технічні характеристики Sun Microsystems для сервлета та JavaServer Pages (JSP). У Tomcat елемент Realm являє собою "базу даних" імен користувачів, паролів і ролей (подібних до груп Unix), призначених для цих користувачів. Різні реалії Realm дозволяють інтегрувати Catalina в оточення, де така інформація про автентифікацію вже створюється та підтримується, а потім використовувати цю інформацію для реалізації керованої безпеки контейнера.

Coyote - це компонент-коннектор для Tomcat, який підтримує протокол HTTP 1.1 як веб-сервер. Це дозволяє Catalina, також виконувати роль простого веб-сервера, який обслуговує локальні файли як HTTP-документи. Coyote слухає вхідні з'єднання з сервером на конкретному порту TCP і пересилає запит до Tomcat Engine для обробки запиту та відправки назад запитуючому клієнту.

Jasper - це Tomcat JSP Engine. Jasper парсить JSP файли для компіляції їх в Java код, як сервлети (які можуть бути оброблені з допомогою Catalina).

Під час виконання програми Jasper виявляє зміни до файлів JSP і перекомпилює їх, якщо такі були. Під час перекомпіляції модифікованого JSP-коду Java, стара версія все ще доступна для запитів сервера. Головний сервлет JSP видаляється після завершення перекомпилювання нового сервлету JSP.

Поєднання бібліотеки тегів JSP - кожна розмітка тегів у файлі JSP обробляється класом обробника тегів. Об'єкти класу обробника тегів можуть бути об'єднані та повторно використані в всьому сервлеті JSP.

Інструмент розробки Java Enterprise Edition пропонує можливісь створювати, так звані “Custom Tag”, тобто теги власноручної розробки, та їх впровадження у систему.

Кластер використовується для керування великими програмами. Він використовується для балансування навантаження, що може бути досягнуто за допомогою багатьох методів. Підтримка кластерингу вимагає JDK версії 1.5 або пізнішої версії.

Функція високої доступності використовується для полегшення планування оновлень системи (наприклад, нових випусків, запитів на зміну), не впливаючи на життя середовища. Це відбувається шляхом відправлення запитів на інший порт тимчасового сервера, тоді як головний сервер оновлюється на основному порту. Це дуже корисно для обробки запитів користувачів на веб-додатках з високим трафіком.

Веб-додатки доповняють як користувальницькі, так і системні веб-додатки, щоб додати підтримку розгортання веб-серверу в різних середовищах. Він також намагається керувати сесіями, а також програмами по всій мережі.

Tomcat розробляє додаткові компоненти. Ряд додаткових компонентів може бути використаний з Apache Tomcat. Ці компоненти можуть бути побудовані користувачами, якщо вони їм потрібні, або їх можна завантажити глобального репозиторія Apache Fundation.

3.6. MySQL Server

MySQL - найпопулярніша у світі база даних з відкритим кодом. Завдяки своїй перевіреній продуктивності, надійності та простоті використання, MySQL стала провідним вибором баз даних для веб-застосунків, що використовуються високопрофесійними веб-ресурсами, включаючи Facebook, Twitter, YouTube, Yahoo! і багато інших.

MySQL - це система управління базами даних. У реляційних базах даних дані зберігаються не всі разом, а в окремих таблицях, завдяки чому можна досягти підвищення швидкості і гнучкості. Таблиці мають зв'язуються між собою за допомогою відносин, завдяки чому забезпечується можливість об'єднувати при виконанні запиту дані з декількох таблиць. SQL як частина системи MySQL можна охарактеризувати як мову структурованих запитів плюс найбільш поширений стандартний мова, яка використовується для доступу до баз даних.

MySQL є системою клієнт-сервер, яка містить багато-SQL-сервер, що забезпечує підтримку різних обчислювальних машин баз даних, а також кілька різних клієнтських програм і бібліотек, засоби адміністрування і широкий спектр програмних інтерфейсів (API). Система безпеки заснована на привілеях та паролях з можливістю верифікації з віддаленого комп'ютера, за рахунок чого забезпечується гнучкість і безпека. Паролі при передачі по мережі при з'єднанні з сервером шифруються. Клієнти можуть з'єднуватися з MySQL, використовуючи сокети TCP / IP, сокети Unix або іменовані канали.

Cтруктура MySQL трирівнева: бази даних - таблиці - записи. Логічно - таблиця являє собою сукупність записів. А записи - це сукупність полів різного типу. Ім'я бази даних MySQL унікально в межах системи, а таблиці - в межах бази даних, поля - в межах таблиці. Один сервер MySQL може підтримувати відразу кілька баз даних, доступ до яких може розмежовуватися логіном і паролем. Знаючи ці логін і пароль, можна працювати з конкретною базою даних. Наприклад, можна створити або видалити в ній таблицю, додати записи і т.д. Зазвичай ім'я-ідентифікатор і пароль призначаються хостинг провайдерами, які і забезпечують підтримку самої бази даних.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3. РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕСПЕЧЕННЯ 28  3.1. Java Virtual Machine 28  3.2. Java cервлет 28  3.3 JavaServer Page. 29  3.4 Веб-контейнер. 30  3.5. Apache Tomcat 33  3.6. MySQL Server 35  5.1.Шаблони проектування  5.2. DAO+DTO  5.3.JDBC  5.4.Prepared statements  5.5.Connection Pool  5.6.Services  5.7.Maven  2.4. MySQL Connector/J 23  2.5. Хеш-фунції SHA-256 23  2.6. Утиліта для ведення журналів Log4j 26  2.7. JSTL 26  1.2.4. Використання транзакцій 17  ВИСНОВКИ  СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ | | | | | | |
|  |  |  |  |  | ДП.468300.004 ПЗ | Арк. |
|  |  |  |  |  | 2 |
| Змін. | Арк. | № докум. | Підпис | Дата |

Розробка архітектури об’єктів доступу до бази даний на основі шаблону проектування DAO

DAO – Data Access Object (об’єкт доступу до даних) абстрагує і інкапсулює доступ до джерела даних. DAO управляє з'єднанням з джерелом даних для отримання та запису даних.

DAO реалізує необхідний для роботи з джерелом даних механізм доступу. DAO повністю приховує деталі реалізації джерела даних від клієнтів. Оскільки при змінах реалізації джерела даних представлений DAO інтерфейс не змінюється, цей патерн дає можливість DAO приймати різні схеми сховищ без впливу на бізнес-компоненти. По суті, DAO виконує функцію адаптера між компонентом і джерелом даних.

При розробці