МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СПЕЦІАЛІЗОВАНИХ КОМП’ЮТЕРНИХ СИСТЕМ

**ЗВІТ**

**з переддипломної практики**

студента VI курсу

групи КВ-63М

шифр КВ-6118

Шолоха О. О.

|  |  |
| --- | --- |
| Керівник переддипломної практики від інституту  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  /Плахотний М.В./  Представлено до захисту  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2012р. | Керівник переддипломної практики від наукової організації  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  /Петрашенко А. В./  Захищено з оцінкою  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2012р. |

Київ

2012р.

**ЗМІСТ**

1. [ПРИЗНАЧЕННЯ ТА ЗАВДАННЯ НАУКОВО-ДОСЛІДНОЇ ПРАКТИКИ](#_Toc317432060)
2. [ХАРКТЕРИСТИКА ПІДПРИЄМСТВА, ДЕ ВІДБУВАЛАСЬ НАУКОВО-ДОСЛІДНА ПРАКТИКА](#_Toc317432059)
3. [ТЕМА ДИПЛОМНОЇ РОБОТИ. ПОСТАНОВКА ЗВДАННЯ. АНАЛІЗ ЗАВДАННЯ](#_Toc317432061)
4. [МЕТОДИ РІШЕННЯ ТА ЇХ ОБҐРУНТУВАННЯ](#_Toc317432062)
5. [АНОТАЦІЇ](#_Toc317432062)
6. [ЗМІСТ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ](#_Toc317432062)
7. [ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕНЬ. ПОРІВНЯЛЬНА ХАРКТЕРИСТИКА МЕТОДІВ ТА ЗАСОБІВ РІШЕННЯ ПРОБЛЕМИ](#_Toc317432062)
8. [СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ](#_Toc317432065)
9. [РОБОЧІ МАТЕРІАЛИ.](#_Toc317432065)

**ПРИЗНАЧЕННЯ ТА ЗАВДАННЯ НАУКОВО-ДОСЛІДНОЇ ПРАКТИКИ**

Переддипломна практика є одним із завершальних етапів підготовки магістра в вищій школі і підготовкою до дипломного проектування. Фактично переддипломна практика є початком виконання дисертації.

Метою і задачами переддипломної практики є:

* поглиблене вивчення і закріплення теоретичних знань, отриманих студентами у вузі, підготовка до самостійної, пошукової, наукової, дослідницької роботи в галузі комп’ютерних систем та мереж;
* надання студентам можливості одержання навичок і уміння аналізувати функціональні залежності, що покладені в основу конкретних технологічних процесів, систем, пов'язаних з використанням сучасних комп’ютерних засобів та мереж;
* поглиблення вміння самостійно працювати з навчальною і науково-методичною базою;
* закріплення навичок оформлення науково-дослідних робіт;
* формування розгорнутого завдання на магістерську дисертацію на основі зібраних і узагальнених у період практики даних;

Зміст та послідовність переддипломної практики визначаються темою магістерської дисертації. В зв'язку з цим кожен студент має свій індивідуальний графік проходження практики. Магістерська робота містить висунуті автором для прилюдного захисту науково обґрунтовані теоретичні та експериментальні результати, наукові положення.

Практична підготовка студентів здійснюється при взаємодії з підприємствами та науковими організаціями з метою вироблення конкретних навичок та вмінь у майбутнього наукового спеціаліста і взаємної зацікавленості в частині підготовки фахівців з мінімальним часом адаптації після закінчення вузу.

**ХАРКТЕРИСТИКА ПІДПРИЄМСТВА, ДЕ ВІДБУВАЛАСЬ НАУКОВО-ДОСЛІДНА ПРАКТИКА**

Факультет прикладної математики НТУУ «КПІ» засновано в 1990 р. з метою підготовки фахівців для тих напрямків життєдіяльності суспільства, що визнані світовою громадськістю як визначальні для початку постіндустріального ХХІ століття. Одним із цих напрямків є математизація науки, техніки, виробництва, суспільних відносин, що полягає в їх формалізації, структуризації й найбільш широкому застосуванні математичних методів розв’язування фізичних, технічних, інформаційних, управлінських та інших задач. Іншим важливим напрямом життєдіяльності є інформатизація суспільного життя, основними інструментальними засобами якої є комп’ютерна та програмна інженерія. При підготовці на факультеті фахівців з цих важливих напрямів використовуються новітні досягнення в галузі фундаментальних та прикладних наук.

Факультет готує фахiвцiв у галузі iнформацiйних технологій, проектування операційних систем, розробки системного та прикладного математичного забезпечення, систем автоматизації проектування, наукових дослiджень, експертних систем, розробки та застосування комп'ютерних систем i мереж загального призначення, спецiалiзованих комп'ютерних систем та мереж з оптимізованими параметрами, засобів захисту iнформацiї в комп'ютерних системах, локальних та розосереджених обчислювальних системах.

Випускники факультету працюють в установах НАН України, галузевих науково-дослідних закладах, органiзацiях та на підприємствах, що займаються розробкою та впровадженням математичних методів, комп'ютерних систем, а також інших програмно-технічних засобiв автоматизацiї iнформацiйних, виробничих та проектно-дослiдних процесів, забезпеченням iнформацiйних технологій, захисту iнформацiї тощо.

**ТЕМА ДИПЛОМНОЇ РОБОТИ. ПОСТАНОВКА ЗВДАННЯ. АНАЛІЗ ЗАВДАННЯ**

Тема мого дипломного проекту: «Розподілені програмні засоби моніторингу Web-ресурсів»

В даній роботі запропонований новий метод розподілення навантаження під час моніторингу Web-ресурсів. Основна ідея даного методу полягає у застосуванні обчислювальних потужностей браузерів клієнтів певних Web-ресурсів.

Розробити розподілену систему моніторингу Web-ресурсів з використанням з застосуванням обчислювальних потужностей браузерів.

Дана система повинна являти собою динамічну Web-сторінку з спеціальним javascript кодом, що може приймати на вхід HTML код сторінки, що підлягають моніторингу, а на вихід згруповані дані та посилання на інші сторінки, що були знайдені на даній.

На сьогоднішній день web-ресурси зайняли домінуюче місце в розповсюдженні інформації. Щосекунди користувачі мережі Інтернет, шукають тематичну інформацію (новини, документацію, наукові статті тощо). Кількість інформації, неперервно збільшується, що в нелінійній залежності збільшує навантаження на сервери, які займаються індексацією цієї інформації.

Web-ресурси по своїй структурі являють собою набір web-сторінок. Web-сторінка в свою чергу є набором інформації і може містити гіпертекст з навігаційними гіперпосиланнями на інші web-сторінки різних web-ресурсів. Така організація web-ресурсів створює можливість розглядати їх як павутину(граф), пройшовшись по вершинах(web-сторінках) якої, ми будемо знати за яким посланням буде знаходитись та чи інша інформація. Проблема полягає в тому, що загальна кількість web-ресурсів є дуже великою і для індексації такої кількості web-сторінок потрібні дуже великі обчислювальні та мережеві потужності.

Останні роки популярності Web-додатків почала значно зростати. Перегляд електронної почти, фото, відео, покупки товарів, спілкування з друзями та колегами переходить з великого різноманіття окремих спеціалізованих додатків до одного – Web-браузера.

Користувачі проводять години часу користуючись цим додатком. При цьому обчислювальна потужність їхніх комп’ютерів в більшості випадків ледве використовується на половину.

Сучасні підходи для асинхронної передачі даних(Ajax) дають можливість за допомогою спеціальних скриптів на мові програмування javascript, яка підтримується більшістю браузерів легко використовувати залишкову потужність.

Основною задачею даної практики є розробка спеціалізованого тестового програмного забезпечення, що дозволяє перевірити ефективність запропонованого в дипломні проекті методу розподілення навантаження під час моніторингу Web-ресурсів.

Слід зазначити наступні задачі та етапи розробки програмного забезпечення для тестування даного методу:

* Проектування загальної архітектури системи;
* Розробка серверної частини системи:
* Проектування бази даних;
* Розробка модулю завантаження HTML коду сторінок, що підлягають моніторингу;
* Розробка модулю збереження отриманих результатів;
* Розробка клієнтної частини системи:
  + Розробка модулю прийму та передачі даних;
  + Розробка модулю аналізу інформації HTML коду сторінки, що підлягає моніторингу;
  + Розробка модулю аналізу посилань з HTML коду сторінки, що підлягає моніторингу.

**МЕТОДИ РІШЕННЯ ТА ЇХ ОБҐРУНТУВАННЯ**

Використання асинхронних запитів надає нам доступ до обчислювальних потужностей клієнта(донора) без створення додаткових незручностей. Користувач може просто відкрити якусь сторінку з цікавою для нього інформацією і ознайомлюватись з нею, і в це й ще час його комп’ютер буде проводити моніторинг інших інформаційних Web-ресурсів.

З переконань забезпечення безпеки в браузерах, вони забороняють відправляти ajax запити на ресурси з відмінним доменом від того на якому знаходиться користувач. В зв’язку з цим ми повинні створити спеціалізований серверний модуль по завантаженню та передачі клієнту HTML коду ресурсів, що підлягають моніторингу.

**АНОТАЦІЇ**

Анотація

В даній роботі пропонуються новий метод організації системи розподіленого моніторингу Web-ресурсів. Робота містить в собі опис методу, особливостей його реалізації та результати проведених досліджень на тестовому програмному забезпеченні.

В основі даного методу лежить ідея використання обчислювальних потужностей браузерів клієнтів. Організація передачі даних повинна здійснюватись за допомого асинхронних (ajax) запитів до спеціалізованого серверу. Такий підхід не створює додаткових незручностей для клієнта, таких як інсталяція додаткового програмного забезпечення чи використання додаткових мережевих портів.

Дослідження проведені на тестовій системі показали позитивні результати. Це дає нам можливість з упевненістю стверджувати, що даний метод є ефективним.

Abstract

This paper proposed a new method of distributed systems for monitoring Web-resources. The work contains a description of the method, the characteristics of its implementation and results of studies on beta software.

The basis of this method is the idea of ​​using computational capabilities of browsers clients. Organization of data should be carried out by auxiliary asynchronous (ajax) requests to a special server. This approach does not create additional inconvenience for the customer, such as installing additional software or use additional network ports.

Research conducted on the test system showed positive results. This enables us to confidently assert that this method is effective.

Аннотация

В данной работе предлагаются новый метод организации системы распределенного мониторинга Web-ресурсов. Работа включает в себя описание метода, особенностей его реализации и результаты проведенных исследований на тестовом программном обеспечении.

В основе данного метода лежит идея использования вычислительных мощностей браузеров клиентов. Организация передачи данных должна осуществляться с работодателем асинхронных (ajax) запросов к специализированному серверу. Такой подход не создает дополнительных неудобств для клиента, таких как установка дополнительного программного обеспечения или использования дополнительных сетевых портов.

Исследования, проведенные на тестовой системе показали положительные результаты. Это дает нам возможность с уверенностью утверждать, что данный метод является эффективным.

**ЗМІСТ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ**

ЗМІСТ

РОЗДІЛ 1: ОГЛЯД ІСНЮЧИХ ЗАСОБІВ ТА МЕТОДІВ МОНІТОРИНГУ WEB-РЕСУРСІВ

1.1 ЗАГАЛЬНІ ВДОМОСТІ ПРО МОНІТОРИНГ WEB-РЕСУРСІВ

1.2 ПЕРВАГИ ТА НЕДОСТАТКИ ІСНУЮЧИХ ЗАСОБІВ ТА МЕТОДІВ МОНІТОРИНГУ WEB-РЕСУРСІВ

ВИСНОВКИ

РОЗДІЛ 2: ОГЛЯД ОСОБЛИВОСТЕЙ ПЕРЕДАЧІ ДАНИХ ДЛЯ МОНІТОРИНГУ ВЕБ РЕСУРСІВ

2.1 ХАРАКТЕРИСТИКА ТА ОСОБЛИВОСТІ ТЕКТОВИХ ПРОТОКОЛІВ ПЕРЕДАЧІ ДАНИХ

2.2 ХАРАКТЕРИСТИКА ТА ОСОБЛИВОСТІ РОБОТО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПЕРЕГЛЯДУ WEB-РЕСУРСІВ

ВИСНОВКИ

РОЗДІЛ 3: РЕАЛІЗАЦІЯ МЕТОДУ РОЗПОДІЛЕНОГО МОНІТОРИНГУ WEB-РЕСУРСІВ З ВИКОРИСТАННЯМ ОБЧИСЛЮВАЛЬНИХ ПОТУЖНОСТЕЙ БРАУЗЕРІВ КЛІЄНТІВ

3.1 ЗАГАЛЬНІ ЕТАПИ РЕАЛІЗАЦІЇ ДАНОГО МЕТОДУ

3.2 АЛГОРИТМ РОБОТИ СИСТЕМИ

3. 3 АЛГОРИТМ РОЗБОРУ КОНТТЕНТУ WEB-РЕСУРСУ

ВИСНОВКИ

РОЗДІЛ 4: ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МЕТОДУ РОЗПОДІЛЕНОГО МОНІТОРИНГУ WEB-РЕСУРСІВ З ВИКОРИСТАННЯМ ОБЧИСЛЮВАЛЬНИХ ПОТУЖНОСТЕЙ БРАУЗЕРІВ КЛІЄНТІВ

4.1 СТВОРЕННЯ МЕТОДИКИ ОЦІНКИ РЕЗУЛЬТАТІВ РОБОТИ МЕТОДУ

4.2 АНАЛІЗ ОБЄМІВ WEB-РЕСУРСІВ, НАД ЯКИМИ МОЖНА ПРОВЕСТИ МОНІТОРИНГ ЗА ДОПОМОГОЮ АНАЛОГІВ ДАНОМУ МЕТОДУ

4.3 АНАЛІЗ ОБЄМІВ WEB-РЕСУРСІВ, НАД ЯКИМИ МОЖНА ПРОВЕСТИ МОНІТОРИНГ ЗА ДОПОМОГОЮ АНАЛОГІВ ДАНОМУ МЕТОДУ

4.4 ПОРІВНЯЛЬНИЙ АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

ВИСНОВКИ

ЗАКІНЧЕННЯ

СПИСОК ЛІТЕРАТУРИ

ДОДАТКИ

/ Петрашенко А. В. / \_\_\_\_\_\_\_\_\_\_ \_\_ лютого 2012

**ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕНЬ. ПОРІВНЯЛЬНА ХАРКТЕРИСТИКА МЕТОДІВ ТА ЗАСОБІВ РІШЕННЯ ПРОБЛЕМИ**

Предметом дослідження є моніторинг окремих тематичних ресурсів Всесвітної павутини.

Всесвітня павутина (англ. World Wide Web, скорочено: WWW) — найбільше всесвітнє багатомовне сховище інформації в електронному вигляді: десятки мільйонів пов'язаних між собою документів, що розташовані на комп'ютерах, розміщених на всій земній кулі. Вважається найпопулярнішою і найцікавішою службою мережі Інтернет, яка дозволяє отримувати доступ до інформації незалежно від місця її розташування.

Основні особливості

В наш час, найбільш бурхливо прогресуюча компонента мережі Інтернет. Користувачі автоматично переходять від однієї бази даних (сайту) до іншої за допомогою гіперпосилань. Кількість серверів WWW постійно зростає, а швидкість росту WWW навіть більша ніж у самої мережі Internet. WWW – найрозвиненіша технологія Internet, вона вже стала масовою. Перспективи розвитку – необмежені.

WWW – інформаційна система, якій не можна дати конкретного визначення. Наведемо лише деякі з епітетів, якими вона може бути позначена: гіпертекстова, гіпермедійна, розподілена, інтегруюча, глобальна. Нижче буде показано, що слід розуміти під кожною з цих властивостей у контексті WWW.

Принцип роботи

WWW працює за принципом клієнт-сервер: існує велика кількість серверів, які за запитом клієнта надають йому гіпермедійний документ. Такий документ складається із частин з різним представленням інформації (текст, графіка, звук, відео, тривимірні об’єкти тощо). В ньому кожен елемент може бути посиланням на інший документ чи його частину. Такі посилання в WWW організовані так, що кожний інформаційний ресурс в глобальній мережі Internet однозначно адресується, а надісланий сервером документ може посилатися на інші документи на цьому ж сервері, чи на документи на інших комп’ютерах Internet. При цьому користувач не помічає цього і працює з усім інформаційним простором Internet, як з єдиним цілим. Посилання WWW вказують не тільки на документи, специфічні для самої WWW, але й на інші сервіси і інформаційні ресурси Internet. Більш того, більшість програм клієнтів WWW (браузер) не просто розуміють такі посилання, а є додатково програмами-клієнтами відповідних сервісів: FTP, Gopher, новин мережі Usenet, електронної пошти і т. і. Таким чином, програмні засоби WWW — універсальні для різних сервісів Internet, а сама інформаційна система WWW грає інтегруючу роль.

Моніторинг Web-ресурсів — це комплекс операцій з пошуку, аналізу, класифікації, кластеризації інформації з Інтернету за тематикою, що цікавить користувача. Таким чином поняття «моніторинг Web-ресурсів» відноситься до моніторингу інтернет-ресурсів, на відміну від поняття «моніторинг мережі», яке відноситься до задач управління мережею. Моніторинг Інтернету дозволяє відстежувати і спостерігати за розвитком тієї або іншої події або теми. З деяких пір моніторинг інтернет-ресурсів став однією з ключових складових діяльності маркетингових служб, інформаційних агентств і служб новин.

Основні методи та інструменти моніторингу інтернет-ресурсів

Існують такі основні методи та інструменти:

* обробка вибраних сайтів у ручному режимі
* пошукові системи і каталоги
* новинна підписка (зокрема RSS)
* спеціалізовані сервіси і програмні продукти моніторингу Інтернету.

Ручна робота

Ручна робота, заснована на регулярній перевірці декількох сайтів, вибраних за заданою темою, - найпростіший і на жаль, не самий ефективний спосіб. Одна людина здатна обробити впродовж доби не більше декількох сотень сторінок, якщо працюватиме без відпочинку і відвернення на інші заняття. Крім того, потрібний час для обходу необхідних сторінок і збереження їх у виділених каталогах.

Пошукові системи і каталоги - здавалося б, також доступний і простий спосіб. Але слід пам'ятати, що будь-які, навіть найбільш популярні, пошукові системи покривають лише частину веб-простору. Час, відведений для індексування нової інформації, може складати від декількох годинників до декількох днів. Для переважної частини бізнес-задач це недозволенно великий час затримки. Крім того, в пошукових системах практично відсутні архіви документів, які колись були виставлені в Інтернеті, але потім видалені. Пошукові системи рідко дозволяють визначити "новизну" того або іншого документа. Нарешті, відкриті пошукові системи рідко індексують ті електронні версії газет, журналів і телепередач, які розташовані на нижніх рівнях відповідних сайтів. Тому часто пропускається велика частина інформативних документів. Використання веб-каталогів веде до тих самих проблем, які пов'язані з ручною обробкою інформації.

Переваги систем моніторингу інтернет-ресурсів у порівнянні з традиційними пошуковими системами

* Оперативність - бази даних таких систем поповнюються кожні 10-20 хвилин, джерела скануються з Інтернету по мірі їх оновлення, тоді як період індексації традиційних пошукових систем може вимірюватися добами.
* Доступність ретроспективного фонду - навіть якщо інформація видалена з веб-сайту джерела, вона збережена в інформаційному сховищі.
* Наявність аналітичного інструментарію - користувач може в режимі реального часу не тільки отримувати результати пошуку, але і формувати дайджести, будувати сюжетні ланцюжки, аналізувати динаміку появи понять тощо.
* Можливість селекції дублікаті - автоматичне маркування ідентичної за змістом новинної інформації.
* Наявність інструментарію багаторівневого уточнення запиту.

Переваги систем моніторингу інтернет-ресурсів у порівнянні з новинними веб-сайтами

* Охоплення джерел - користувач має доступ до новин з тематики, що цікавить його, одночасно з великої кількості веб-сайтів, включаючи і ті вибрані, які він звик проглядати щодня.
* Принцип "одних рук" - користувач системи моніторингу має доступ до інформації з багатьох веб-сайтов з одного інтуїтивного інтерфейсу.
* Пошукові можливості - новинні веб-сайти, на відміну від систем моніторингу, не завжди мають розвинені пошукові можливості.
* Доступність ретроспективного фонду.

Професійне використання можливостей системи моніторингу інтернет-ресурсів забезпечує якісно нові можливості для інформаційно-аналітичної роботи.

Пошукова система — онлайн-служба (програмно-апаратний комплекс з веб-інтерфейсом), що надає можливість пошуку інформації в Інтернеті. У просторіччі під пошуковою системою розуміють веб-сайт, на котрому розміщено інтерфейс (фронт-енд) системи. Програмною частиною пошукової системи є пошукова машина— комплекс програм, що забезпечує функціональність пошукової системи і, зазвичай, є комерційною таємницею компанії-разробника пошукової системи.

Більшість пошукових систем шукають інформацію на сайтах Всесвітньої павутини, але існують також системи, здатні шукати файли на ftp-серверах, товари в інтернет-магазинах, а також інформацію в групах новин Usenet.

Індексація в пошукових системах сайтів здійснюється пошуковим роботом.

Пошуковий робот ("веб-павук", краулер, спайдер) - програма, що є складовою частиною пошукової системи і призначена для обходу сторінок інтернету з метою занесення інформації про них (ключових слів) у базу даних.

Павуки здійснюють загальний пошук інформації в Інтернеті. Вони повідомляють про зміст знайденого документа, індексують його і добувають підсумкову інформацію. Вони також переглядають заголовки, деякі посилання і відправляють проіндексовану інформацію до бази даних пошукового механізму.

AJAX (Asynchronous JavaScript And XML) — підхід до побудови користувацьких інтерфейсів веб-застосувань, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі відправляє запити на сервер і сама звідти довантажує потрібні користувачу дані. AJAX — один з компонентів концепції DHTML.

Про AJAX заговорили після появи в лютому 2005-го року статті Джесі Джеймса Гарретта (Jesse James Garrett) «Новий підхід до веб-застосунків». AJAX — не самостійна технологія. Це ідея.

AJAX — це не самостійна технологія, а швидше концепція використання декількох суміжних технологій. AJAX підхід до розробки призначених для користувача інтерфейсів комбінує кілька основних методів і прийомів:

Використання DHTML для динамічної зміни змісту сторінки.

Використання XMLHttpRequest для звернення до сервера «на льоту», не перезавантажуючи всю сторінку повністю

* альтернативний метод — динамічне підвантаження коду JavaScript в тег <SCRIPT> з використанням DOM, що здійснюється із використанням формату JSON)
* динамічне створення дочірніх фреймів

Використання цих підходів дозволяє створювати набагато зручніші веб-інтерфейси користувача на тих сторінках сайтів, де необхідна активна взаємодія з користувачем. AJAX — асинхронний, тому користувач може переглядати далі контент сайту, поки сервер все ще обробляє запит. Браузер не перезавантажує web-сторінку і дані посилаються на сервер без візуального підтвердження (крім випадків, коли ми самі захочемо показати процес з’єднання з сервером). Використання AJAX стало найбільш популярне після того, як компанія Google почала активно використовувати його при створенні своїх сайтів, таких як Gmail, Google Maps і Google Suggest. Створення цих сайтів підтвердило ефективність використання даного підходу.

Порівняння класичного підходу та AJAX

Класична модель веб-застосування:

* Користувач заходить на веб-сторінку і натискає на який-небудь її елемент
* Браузер надсилає запит серверу
* У відповідь сервер генерує повністю нову веб-сторінку і відправляє її браузеру і т. д.

З боку сервера можлива генерація не всієї сторінки наново, а тільки деяких її частин, з подальшою передачею користувачу.

Модель AJAX:

* Користувач заходить на веб-сторінку і натискає на який-небудь її елемент.
* Браузер відправляє відповідний запит на сервер.
* Сервер віддає тільки ту частину документа, яка змінилася.

Варіації

В деяких застосуваннях використовуються певні варіації з форматом відповіді сервера, такі варіації набули напівофіційні назви.

AHAH (Asynchronous HTML and HTTP) — це споріднений AJAX підхід для динамічного оновлення веб-сторінок, використовуючи JavaScript. Основною його відмінністю від AJAX є те, що відповіді сервера повинні бути звичайним HTML.[1] Перевага підходу полягає в більшій сумісності і функціональності (підтримка навігаційних кнопок браузера, аплоад файлів тощо). Реалізується у вигляді звичайних фреймів, що автоматично міняють свій розмір під розмір вмісту, або у вигляді прихованих фреймів, що виконують тільки функції завантаження даних.

Asynchronous XHTML and HTTP, або абревіатура AXAH — це майже те ж саме що і AHAH. Різниця тільки в тому, що в AHAH сервер клієнтові повертає HTML, а в AXAH вже XHTML.

Браузери, які підтримують AJAX

Зауважте, що це загальний список і підтримка застосувань AJAX залежатиме від особливостей підтримки оглядача:

Версія Microsoft Internet Explorer 5.0 і вище, а також оглядачі, засновані на ньому (Версії ОС Mac не підтримуються)

Оглядач Opera та програмне забезпечення для пристроїв від Opera Software.

Засновані на Gecko подібно до Mozilla, Mozilla Firefox, SeaMonkey, Camino, Flock, Epiphany, Galeon та версія Netscape 7.1 і вище.

Оглядачі з вбудованим KHTML API (WebKit) версії 3.2 і вище, зокрема Konqueror 3.2 і вище, версія Apple Safari 1.2 і вище, Google Chrome.

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Рзаєв Г.І. Використання Інтернет-ресурсів у менеджменті конкурентоспроможності підприємств // Вісник Національного університету "Львівська політехніка". Менеджмент та підприємництво в Україні: етапи становлення і проблеми розвитку, 2009. - T 647. - C. 596-600.
2. Ландэ Д.В. Поиск знаний в Internet. - М.: Диалектика, 2005. - 272 c.
3. Григорьев А.Н., Ландэ Д.В., Бороденков С.А., Мазуркевич Р.В., Пацьора В.Н. InfoStream. Мониторинг новостей из Интернет: технология, система, сервис: научно-методическое пособие.- Киев: ООО "Старт-98", 2007. - 40 с.
4. Колисниченко Д. Н. Поисковые системы и продвижение сайтов в Интернете — М.: Диалектика, 2007. — 272 с.
5. Ашманов И. С., Иванов А. А. Продвижение сайта в поисковых системах — М.: Вильямс, 2007. — 304 с.
6. Байков Владимир Дмитриевич Интернет. Поиск информации. Продвижение сайтов — СПб.: БХВ-Петербург, 2000. — 288 с.
7. Маннинг К., Рагхаван П., Шютце Х. Введение в информационный поиск — Вильямс, 2011.
8. Петренко А.І. Грід - системи з розробки та оптимізації інженерних рішень.- Системний аналіз та інформаційні технології: 13-та міжнародна науково-технічна конференція “САІТ-2011”, 25-29 травня 2011 р., м.Київ, Україна: матеріали. - К.: ННК “ІПСА” НТУУ “КПІ”, 2011.
9. http://en.wikipedia.org/wiki/Web\_search\_engine
10. http://en.wikipedia.org/wiki/URL\_normalization
11. http://en.wikipedia.org/wiki/Uniform\_Resource\_Locator
12. http://tools.ietf.org/html/rfc3986
13. http://tools.ietf.org/html/rfc1866
14. http://en.wikipedia.org/wiki/Web\_browser
15. http://en.wikipedia.org/wiki/Ajax
16. http://www.ietf.org/rfc/rfc2119.txt
17. http://grid.kpi.ua/
18. Згуровский М.З., Петренко А.І. Е-наука на шляху до семантичного Грід. Частина 1: Об’єднання Web- і Грід- технологій.- //Системні дослідження і інформаційні технології.-Київ, №1, 2010.-с.26-38.
19. Киселев Г.Д. Мониторинг мультисервисных компьютерных сетей средствами системы Nagios / Киселев Г.Д., Шпакаускас М.С. // Электроника и связь : тематический выпуск «Электроника и нанотехнологии». – 2010. – № 3. – С. 178–181.

**РОБОЧІ МАТЕРІАЛИ**.

Додаток 1

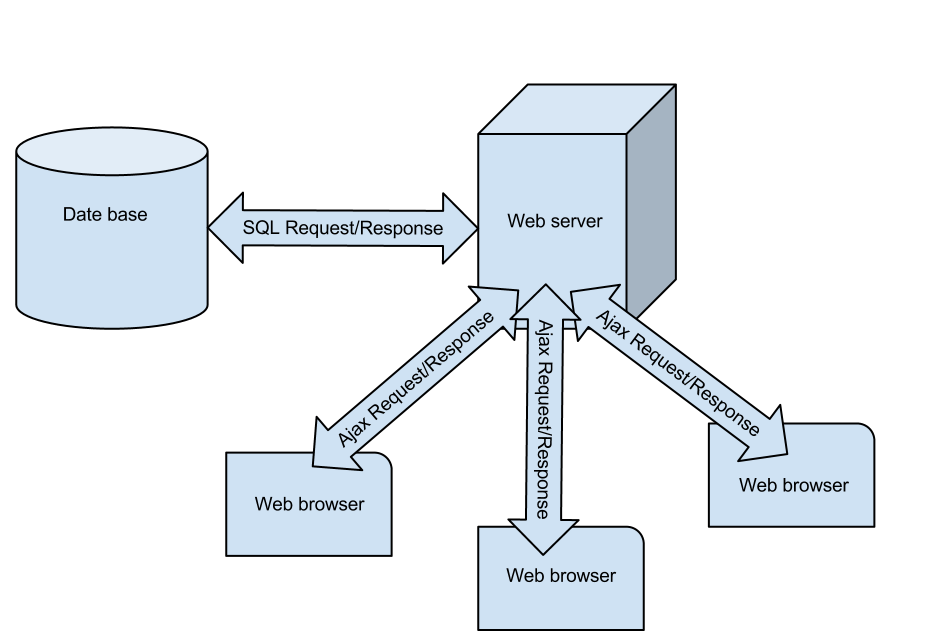


Рис. 1 Структурна схема системи

Додаток 2

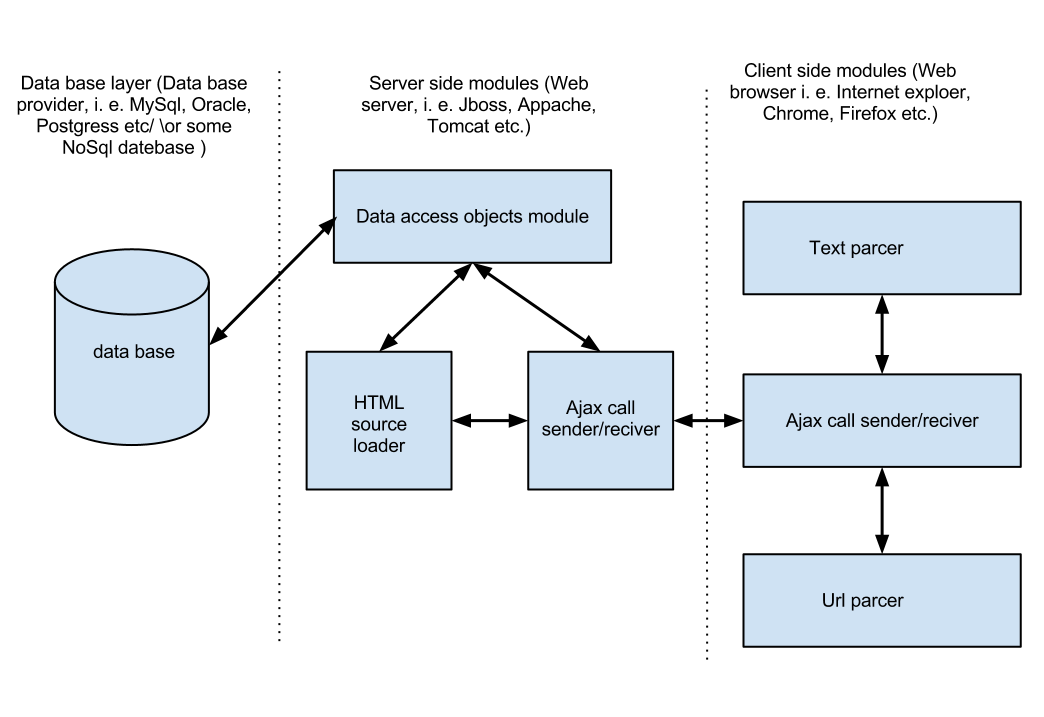


Рис. 1 Схема взаємодії модулів системи

Додаток 3

Програмний код модулів клієнту (javascript)

logiInfo = **true**;

**if** (**typeof** String.prototype.startsWith != 'function') {

String.prototype.startsWith = **function**(str) {

**return** **this**.indexOf(str) == 0;

};

}

**function** setUrl(url) {

$.post("setUrl.htm", {

"urlToDb" : url

}, **function**(data) {

alert("seted");

});

}

**function** getPageUrlTest(pageUrl) {

callTime = **new** Date().getTime();

$.post("getpageByUrl.htm", {

"pageUrl" : pageUrl

}, **function**(data) {

processHTML(data.page);

});

}

**function** startNode() {

callTime = **new** Date().getTime();

$.post("getpage.htm", {}, **function**(data) {

processHTMLAndSendResponse(data.page);

});

}

**function** processHTML(page) {

startProcessingTime = **new** Date().getTime();

$('#bufHtml')[0].innerHTML = "";

page.source = page.source.replace(/<head>.\*<\/head>/i, "");

page.source = page.source.replace(/<link.\*?>/g, "");

page.source = page.source.replace(/<script.\*?\/script>/gi, "");

page.source = page.source.replace(/src=.\*?"/gi, "src=''");

page.source = page.source.replace(/src=.\*?'/gi, "src=''");

page.source = page.source.replace(/style=.\*?'/gi, "src=''");

page.source = page.source.replace(/style=.\*?"/gi, "src=''");

page.source = page.source.replace(/href="javascript:.\*?"/gi, "");

page.source = page.source.replace(/href='javascript:.\*?'/gi, "");

page.source = page.source.replace(/href="skype:.\*?"/gi, "");

page.source = page.source.replace(/href='skype:.\*?'/gi, "");

$('#bufHtml')[0].innerHTML = page.source;

$('#url')[0].value = (page.url);

response = **new** Object();

response.url = page.url;

$("#console")[0].value += "\n\n Url " + page.url;

$("#console")[0].value += "\n Url domain " +page.url.match(/.\*:\/\/.\*?\//i)[0];

urls = getURLs(page.url.match(/.\*:\/\/.\*?\//i)[0]);

response.urls = JSON.stringify(urls);

**if** (logiInfo) {

currentTime = **new** Date();

**var** processingTime = currentTime.getTime()

- startProcessingTime;

**var** totalTime = currentTime.getTime() - callTime;

$("#console")[0].value += "\n\n " + currentTime;

$("#console")[0].value += "\n Request Processed " + page.url;

$("#console")[0].value += " " + urls.length

+ " unical links found.";

$("#console")[0].value += " Time for processing = "

+ processingTime;

$("#console")[0].value += "\n Total time = " + totalTime;

$("textarea")[0].scrollByPages(1);

}

$('#bufHtml')[0].innerHTML = "";

callTime = **new** Date().getTime();

**return** response;

}

**function** processHTMLAndSendResponse(page) {

processHTML(page);

setRespose(response);

}

**function** setRespose(response){

$.post("setresalt.htm", response, **function**(data) {

processHTMLAndSendResponse(data.page);

});

}

**function** getURLs(parent) {

**var** locationUrl = location.href.replace("process.htm", "");

**var** locationDomain = location.origin;

**var** aElements = $("a");

**var** len = aElements.length;

hrefList = **new** Array(len);

**for** ( **var** i = 0; i < len; ++i) {

hrefList[i] = aElements[i].href.replace(/#.\*/, "");

**if** (hrefList[i].startsWith(locationUrl)) {

hrefList[i] = hrefList[i].replace(locationUrl, parent);

} **else** **if** (hrefList[i].startsWith(locationDomain)) {

hrefList[i] = hrefList[i].replace(locationDomain + "/", parent);

}

}

**return** jQuery.unique(hrefList);

}

Додаток 4

Програмний код модулів серверу (java)

**package** ua.kpi.scs.controller;

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.Map;

**import** javax.servlet.http.HttpServletRequest;

**import** org.apache.log4j.Logger;

**import** org.json.JSONArray;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.ResponseBody;

**import** ua.kpi.model.PageResponse;

**import** ua.kpi.service.DbPageService;

@Controller

**public** **class** GridController {

**private** **static** **final** Logger *LOGGER* = Logger.*getLogger*(GridController.**class**);

**private** DbPageService dbPageService;

@RequestMapping(value = "process.htm")

**public** String showPage(**final** HttpServletRequest request) {

**return** "process";

}

@RequestMapping(value = "setUrl.htm")

**public** @ResponseBody

**final** Map<String, Object> setUrl(**final** HttpServletRequest request) {

String urlStr = request.getParameter("urlToDb");

**final** Map<String, Object> response = **new** HashMap<String, Object>();

dbPageService.setUrlToDb(urlStr);

**return** response;

}

@RequestMapping(value = "getpage.htm")

**public** @ResponseBody

**final** Map<String, Object> getPage(**final** HttpServletRequest request) {

**final** Map<String, Object> response = **new** HashMap<String, Object>();

response.put("page", dbPageService.getPage());

**return** response;

}

@RequestMapping(value = "getpageByUrl.htm")

**public** @ResponseBody

**final** Map<String, Object> getPageByUrl(**final** HttpServletRequest request) {

String pageUrl = request.getParameter("pageUrl");

**final** Map<String, Object> response = **new** HashMap<String, Object>();

response.put("page", dbPageService.getPagebyUrl(pageUrl));

**return** response;

}

@RequestMapping(value = "setresalt.htm")

**public** @ResponseBody

**final** Map<String, Object> setResponseAndGetNewPage(**final** HttpServletRequest request) {

**final** Map<String, Object> response = **new** HashMap<String, Object>();

PageResponse pageResponse = **new** PageResponse();

pageResponse.setUrl(request.getParameter("url"));

**try** {

**final** JSONArray jsonArray = **new** JSONArray(request.getParameter("urls"));

**final** List<String> urlList = **new** ArrayList<String>();

**int** len = jsonArray.length();

**if** (jsonArray != **null**) {

**for** (**int** i = 0; i < len; i++) {

urlList.add(jsonArray.get(i).toString());

}

}

**final** String[] urls = **new** String[urlList.size()];

**int** i = 0;

**for** (Iterator<String> iterator = urlList.iterator(); iterator.hasNext(); ++i) {

urls[i] = iterator.next().toString();

}

pageResponse.setUrls(urls);

pageResponse.setUrl(request.getParameter("url"));

} **catch** (Exception e) {

*LOGGER*.error("JsonParseException", e);

pageResponse.setUrls(**new** String[0]);

pageResponse.setUrl("STOP");

}

dbPageService.setResponse(pageResponse);

response.put("page", dbPageService.getPage());

**return** response;

}

**public** DbPageService getDbPageService() {

**return** dbPageService;

}

@Autowired

**public** **void** setDbPageService(DbPageService dbPageService) {

**this**.dbPageService = dbPageService;

}

}

package ua.kpi.scs.entities;

import java.io.Serializable;

import java.util.Date;

import javax.persistence.Basic;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.Table;

import javax.persistence.Temporal;

import javax.persistence.TemporalType;

@Entity

@NamedQueries({

@NamedQuery(name = Url.GET\_NOT\_PARSED\_URL, query = "SELECT p FROM Url p WHERE p.parsed = false"),

@NamedQuery(name = Url.GET\_URL\_BY\_ADDRESS, query = "SELECT p FROM Url p WHERE p.url=:url") })

@Table(name = "urls")

public class Url implements Serializable {

public static final String GET\_NOT\_PARSED\_URL = "notParced";

public static final String GET\_URL\_BY\_ADDRESS = "urlByAddress";

public static final int MAX\_URL\_LENGTH = 255;

private static final long serialVersionUID = 1L;

@Id

@Column(name = "ID")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "CONTENT")

private String content;

@Column(name = "PARENT\_URL\_ID")

private int parentUrlId;

@Column(name = "URL", length = MAX\_URL\_LENGTH, unique = true)

private String url;

@Column(name = "PARSED")

private boolean parsed;

@Basic(optional = false)

@Column(name = "LAST\_UPDATE", columnDefinition = "TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP")

@Temporal(TemporalType.TIMESTAMP)

private Date lastUpdate;

public Url() {

}

public int getId() {

return this.id;

}

public void setId(int id) {

this.id = id;

}

public String getContent() {

return this.content;

}

public void setContent(String content) {

this.content = content;

}

public int getParentUrlId() {

return this.parentUrlId;

}

public void setParentUrlId(int parentUrlId) {

this.parentUrlId = parentUrlId;

}

public String getUrl() {

return this.url;

}

public void setUrl(String url) {

this.url = url;

}

public boolean isParsed() {

return parsed;

}

public void setParsed(boolean parsed) {

this.parsed = parsed;

}

public void setLastUpdate(Date lastUpdate) {

this.lastUpdate = lastUpdate;

}

public Date getLastUpdate() {

return lastUpdate;

}

}

**package** ua.kpi.service;

**import** java.io.BufferedReader;

**import** java.io.InputStreamReader;

**import** java.net.URL;

**import** java.net.URLConnection;

**import** org.apache.log4j.Logger;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

**import** ua.kpi.model.Page;

**import** ua.kpi.model.PageResponse;

**import** ua.kpi.scs.dao.impl.UrlDaoImpl;

**import** ua.kpi.scs.entities.Url;

@Component

**public** **class** DbPageService {

**private** **final** **static** Logger *LOG* = Logger.*getLogger*(DbPageService.**class**);

**private** UrlDaoImpl dao;

**public** Page getPage() {

String pageSource = **null**;

Page page = **new** Page();

**while** ((pageSource == **null**) || (pageSource.length() == 0)) {

page.setUrl(dao.findAddressToParse());

pageSource = getPageHtmlSource(page.getUrl());

}

page.setSource(pageSource);

**return** page;

}

**public** Page getPagebyUrl(**final** String url) {

String pageSource = **null**;

**final** Page page = **new** Page();

page.setUrl(url);

pageSource = getPageHtmlSource(page.getUrl());

page.setSource(pageSource);

**return** page;

}

**public** **void** setUrlToDb(String urlStr) {

Url url = **new** Url();

url.setUrl(urlStr);

dao.save(url);

}

@Autowired

**public** **void** setDao(UrlDaoImpl dao) {

**this**.dao = dao;

}

**public** UrlDaoImpl getDao() {

**return** dao;

}

**public** **void** setResponse(PageResponse pageResponse) {

String[] urls = pageResponse.getUrls();

**for** (**int** i = 0; i < urls.length; i++) {

**if** ((urls[i] == **null**) || (urls[i].length() == 0) || (urls[i].length() > Url.*MAX\_URL\_LENGTH*) || (urls[i] == "")) {

*LOG*.info("Cannot save URL " + urls[i]);

} **else** {

Url url = **new** Url();

url.setUrl(urls[i]);

dao.save(url);

}

}

}

**private** String getPageHtmlSource(String pageUrl)

{

StringBuffer buffer = **new** StringBuffer();

**try** {

URL site = **new** URL(pageUrl);

URLConnection yc = site.openConnection();

BufferedReader in = **new** BufferedReader(**new** InputStreamReader(yc.getInputStream()));

String inputLine;

**while** ((inputLine = in.readLine()) != **null**) {

buffer.append(inputLine);

**if** (buffer.length() > 1000000) {

*LOG*.debug("Page is too long length = " + buffer.length() + ", so we'll ignore it");

**break**;

}

}

in.close();

} **catch** (Exception e) {

*LOG*.debug("Server cannot get page " + pageUrl, e);

}

*LOG*.debug("Length of buffer " + buffer.length());

**return** buffer.toString();

}

}