УДК 000-004.8

О.І. Чумаченко, С.О. Пономаренко

ВИКОРИСТАННЯ СУЧАСНИХ ВЕБ-ТЕХНОЛОГІЙ ДЛЯ ПОБУДОВИ ТЕМАТИЧНОГО ПОШУКОВОГО РОБОТА-КРАУЛЕРА АНАЛІЗАТОРА

Проблема пошуку інформації, займає одне із основних місць в сучасному обзорі розвитку Інтернет простору. Маючи велику і швидко розвиваючу інфраструктуру мереж, неважко розгубитись в тому обсязі даних, що її накопичується з кожним роком. Важко організувати загальну систему, якій було б підпорядковано всі данні про існуючі ресурси та їх розміщення в Інтернеті. Як правило окремі системи охоплюють лише окрему частину цієї інформації, при цьому дані про ресурс що його описують може характеризуватись тільки наявністю тих чи інших даних, і не більше того.

Дуже часто користувачі Інтернету шукають інформацію, більш вузького профілю, актуальність якої інколи не відповідає дійсності. Або зовсім приводить до хибних ресурсів. Можливість надати, саме те, що потрібно, ось що головне в сучасних пошукових системах.

Поставлені завдання під силу так званим роботам-краулерам. Машинам зі задатками штучного інтелекту, що здатні самостійно виконувати пошук інформації у кіберпросторі. Краулер (пошуковий робот, павук) - програма, що є складовою частиною пошуковою системи, головним призначенням котрої є прохід сторінок Інтернету з ціллю занесення інформації про них(ключові слова) до бази пошуковця. По своїй суті такий павук нагадує більш за все звичайний браузер. Він сканує наповнення сторінок, скидає його на сервер пошукової машини, до котрої належить, і відправляється по посиланням на наступні сторінки.

Постановка задачі

Метою проекту є розробка систему пошуку, аналізу та індексації інформації глобальних ресурсів інтернету за спеціалізованими напрямками. Головна задача полягає в розробці системи, що здатна за певними критеріями вказати, наскільки знайдена інформація відповідає обраній тематиці запиту на пошук.

Система представляє з себе робота пошуковця(краулера), головним завданням котрого є аналіз та виділення інформації знайденої на сторінці. Одною з головних характеристик такого робота є його здатність оброблювати інформацію з заданим порогом достовірності, правильності розпізнавання та фільтрування при умові погано сформованих даних сторінок (некоректна та нетипова розмітка HTML сторінок). Черги проходження сторінок, частота візитів, захист від замкнених переходів, а також критерії виділення ключових слів визначаються алгоритмами пошукових машин.

 ISSN 1562-9945

Пошук аналогічних рішень побудови таких систем може нас привести лише до сучас пошукових машин інтернету, розроблених провідними фірмами, як: Google, Yahoo, Yandex, Aport, AltaVista. Кожна з цих систем характеризуються здатність швидко знаходити (раніше проіндексовану) інформацію, згідно пошукового запиту. Достовірність цієї ж інформації як правило залежить від алгоритму системи, і дуже часто опирається на функції підрахунку релевантності сторінки, зовсім не опираючись на зміст тексту.

Розроблювана система, на відміну від представлених вище аналогів, характеризується саме здатністю аналізу, та виділенню потрібної інформації з загальних необроблених даних. Використовуючи таку систему, як другим шаром обробки пошукової інформації, ми зможемо отримати дані що найбільш точно відображають характеристики обраної тематики, тобто пошук може бути спрямований на інформацію більш вузької спеціалізації.

Недоліком же в системі може слугувати тільки її залежність від первинних пошукових систем, що використовуються для отримання посилань на вже існуючі ресурси. При характерних вдосконаленнях, вона здатна самостійно виконувати первинний пошук, проте це може трохи збільшити час на обробку первинної не індексованої інформації. Ця проблема залежить лише від апаратних ресурсів, що будуть виділені під неї.

Функціональна схема такої системи, може бути представлена елементами та зв'язками між ними за допомогою рис. 1.

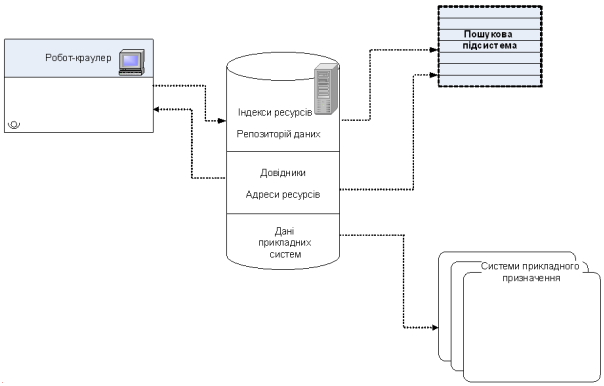

Рис. 1 – Діаграма зв'язків та компонентів пошукової системи

З цієї схеми видно, що краулер виступає окремим автономним модулем(системним процесом), зв'язок котрого з іншими частинами системи здійснюються через базу знань. Саме база знань в цьому випадку виступає центральним елементом. Тому її правильна побудова та встанов-

лення зв'язків між внутрішніми сутностями, може значно спростити і підвищити функціональність системи в цілому.

Структурна схема побудови такої системи основана на функціональній, і також може бути представлена за допомогою рис. 1, а точніше структурними блоками, з яких складається кожен елемент системи. Умовно структуру можна поділити на декілька складових частин, кожна з яких характеризується використанням, свого роду специфічних та інноваційних, технологій та підходів. Унікальні ж комбінації цих складових частин надають можливості в більшій чи меншій мірі використовувати принади таких роботів (потужний пошук, велика здатність розпізнавання та аналізу, швидкодія робота і т.д.). Тож для побудови повноцінного робота-пошуковця необхідно спроектувати та взаємопов'язати наступні складові частини:

- пошуковець актуальної інформації на веб-ресурсах Інтернет простору, згідно обраної тематики запиту
- аналізатор та збирач інформації зі сторінок
- модуль синхронізації даних отриманих з веб-запитів з серверною базою даних

Представивши таким чином функціональну та структурну схеми побудову такої системи, спробуємо дати більш детальний опис її складових частин та технологій, що використовуються для проектування.

Критерії оцінки роботи системи

Для оцінки роботи представленої системи доцільно привести два основних критерії:

- швидкодія
- якість обробки інформації

Перший критерій швидкодії відноситься як до краулера, так і до бази знань. В випадку з краулером даний критерій буде не сильно впливати на швидкодію всієї системи, адже ця частина є автономною, а отже не може впливати на інші частини системи. Ця характеристика в даному випадку буде залежати від апаратний ресурсів, що відведені для підсистеми краулера. Що ж до бази знань, то тут швидкодія відіграю головну роль. Правильна побудова та взаємовідношення окремих сутностей всередині бази знань, ось що буде впливати на цей показник. Пошукова підсистема, є системою реального часу, а отже повинна миттєво реагувати на дії користувача. Правильна побудовані запити та точність вказання всіх його складових дозволять зменшити час очікування відповіді, в свою чергу надлишковість інформації може призвести до виникнення додаткового пошуку по зайвим даним, що в свою чергу призведе до затримок.

Якість обробки, або точність співставлення знайденої інформації обраній тематичній категорії є головним показником робота-краулера. Головною особливістю нашої системи, є саме здатність аналізувати знайдену

інформацію, тож маючи добре побудовані критерії оцінки даних ми з певною вірогідністю зможемо співвіднести їх відповідній тематичній категорії. Правила побудови таких залежностей, як правило будуються на основі семантичного та структурного розгляду певної частини інформації. Кожне правило може бути біль конкретно узагальнене під обрану категорію або під групу категорій.

Маючи критерії оцінки роботи системи, ми можемо сказати на скільки якісно вона працює, та чи має здатність вирішувати поставлені перед нею задачі прикладних систем.

Формалізація

Алгоритмічне забезпечення

Спробуємо розглянути схему функціонування системи виходячи із процесів що протікають в кожній з частин та їх зв'язку між собою.

Модуль пошуку тематичної інформації.

Як більшість пошукових машин, першочерговою задачу краулера є пошук необхідної тематичної інформації. Перш за все необхідно визначитись з тематикою пошуку, та шляхом проходження та знаходження необхідної інформації. Більшість з цих задач було вже поставлено до провідних пошукових систем світу, і кожна з них знайшла свій вихід та основний підхід до вирішення проблем, "як саме шукати?". Головним чином вся ця інформації тримається в секреті фірмами розробника та щільно утримується від сторонніх очей.

Проте проаналізувавши порядок та формат отримання даних з Інтернету, ми можемо спроектувати декілька підходів для роботи нашого пошуковця. По-перше необхідно визначити тематику та направленість нашого робота. В цьому разі, якщо він спрямований сам на пошук інформації тоді даний модуль виділяються в окрему підсистему, що має власну програму та інтерфейс обміну з БД. Така система здатна досліджувати всі актуальні (живі) IP адреси сайтів та порталів, скачувати їх наповнення та передавати дані до робота аналізатора. Позитивні сторони такої реалізації дозволяють нам отримати потужну систему з великим обсягом доступної актуальної інформації з всіх поточних ресурсів Інтернету. Нажаль великим недоліком такої реалізації є необхідність встановлення багатопотокового та ресурсоємнісного сервера з окремим незалежним ядром обробки. Також проходження всього простору Інтернету займе багато часу, і в порівнянні з менш потужною системою аналізу (період обробки значно перевищує період знаходження даних) буде давати накопичувальний ефект на входах та виходах системи.

Однією з найбільш вірогідних реалізацій даної концепції є переведення існуючої бази даних на базу знань, та введення тематичних розділів пошуку даних. В такому разі, замість проходження всі адрес, до робота пошуковця додається модуль генерації доменних імен згідно обраних категорій. В цьому разі пошук інформації здійснюється на основі створених можливих тематичних доменних імені сайтів(порталі), з великою

вірогідністю, при попаданні на котрі ми отримаємо бажану інформацію. Позитивними аспектами при втілені такої програми є отримання більш актуального контенту згідно обраної тематики, час пошуку та проходження Інтернет простору значно зменшується, кількість зайвої інформації також стаю мінімальною. Недоліками є можлива невідповідність знайденої інформації обраному тематичному доменному імені, проте це все може бути легко усунено завдяки коректній роботі робота аналізатора, що відфільтровує знайдені данні по обраним категоріям.

Останнім способом реалізації пошуковця, найбільш точним, за найбільш спрощеною моделлю, це використання заповнених довідників з адресами Інтернет ресурсів, вже віднесених до певних тематичних груп. Кожна адреса що вноситься до цього довідника на відсотків 80 повинна бути оброблена операторам, та підтверджена релевантність інформації, що розташовується на порталі обраній категорії бази знань. Це означає що довідник наповнюється майже весь оператором, або туди вносяться роботом нові веб-адреси, що мають достатній рівень довіри (порогові значення відповідності даних обраній тематиці). Тож робота модуля аналізатора відіграє тут визначальну роль, адже він повинен вирішувати чи буде знайдений ресурс й надалі оглядатись для отримання актуальної інформації чи просто ігноруватись.

Модуль аналізу та збору інформації

Дана частина є основною (ядром) для нашого краулера. Головна задача цього модуля полягаю в знаходженні корисної інформації в переданому потоці отриманого від робота-пошуковця. Загалом концепція побудови цієї частини майже однакова, але все ж таки треба зважати на варіант реалізації робота-пошуковця, адже від того, як багато ми знаємо про те що ми отримали (метадані), залежить наскільки глибоко необхідно аналізувати дану інформацію.

Більш гнучкий, проте, і більш важкий у плані реалізації, варіант передбачає отримання чистого тексту сторінку, тільки з вказанням до якого розділу бази знань відноситься даний ресурс (що, ще також необхідно перевірити). В цьому випадку аналізатор використовує найбільш вживані правила з'ясування релевантності сторінки, тобто актуальності та правдивості інформації. Як правило можна виділити декілька правил, за яким будуть оцінюватись отримані данні, сумарний результат по кожному з яких дасть нам загальну оцінку обраної сторінки:

- Вік сайту.
- Назва URL сайту (ім'я домена).
- Мова сайту.
- Об'єм текстової інформації на веб-сторінці.
- Застосовані стилі до сторінки.
- Загальна кількість ключових слів.
- Індекс цитування.

- Періодичність оновлення інформації на сторінці.
- Кількість графічних та мультимедійних даних на сторінці.
- Використання фреймів.
- Розмір та тип шрифту ключових слів та заголовків.
- Наявність та аналіз мета-тегів.
- Географічне положення сайту.
- Тип сторінки (html або asp).
- Наявність у складі сторінки f lash модулів.
- Наявність "шумових слів".
- Загальна кількість гіперпосилань на внутрішні та зовнішні ресурси.

Даний перелік може бути розширений або скорочений а також за вибором змінюватись для різних категорій. Все залежить від того, на скільки точну або актуальну чи практичну інформацію ми хочемо отримати.

Тож проаналізувавши сторінку згідно обраних правил та зі поставивши отримане значення з пороговим, аналізатор вирішую про необхідність зчитування корисної інформації з сторінку та перенесення її до репозиторію бази знань. Також можливий варіант з внесенням оціненої сторінки до довідника бази знань, зі вказанням відповідного ступеню довіри.

Менш ресурсоємний варіант, передбачає приналежності кожній вебсторінці, шаблону обробки даних (також можливо отриманого при першому проході робота аналізатора, або створеного чи відредагованого оператором). Такий шаблон представляє з себе опис структури сторінки, тобто робота модуля аналізатора в цьому випадку зводиться до мінімуму і все навантаження переходить на збирача. Головної перевагою є швидкодія обробки таких ресурсів, недоліком же є сильна залежність від зміни структури сторінок.

Комбінація з роботу аналізатора для першого проходу з використанням більш гнучкого методу і збирача для аналізу по схемі сторінок, можуть дати найкращий результат при реалізації даного модулю.

Модуль синхронізації даних з базою знань

Система управління базою знань (саме знань, а не даних) повинна забезпечити уявлення і обробку моделі, зіставною по своїй складності з моделлю що використовується свідомістю людини.

Найбільш важливий параметр БЗ — якість знань, що накопичені в ній. Кращі БЗ містять релевантну і свіжу інформацію, мають довершені системи пошуку інформації і дотошно пророблену структуру і формат знань.

Структура і побудова такої БЗ залежить від того, на яку саме з частин вона буде орієнтована. БЗ може буди розроблена під структуру краулера, тобто вона більш абстрактна і всі типи, формати та дані зберігаються у репозиторії з рекурсивними зв'язками. Така архітектура повинна обов'язково складатись з таблиць основних даних, та додаткових (описових) таблиць з метаданими.

Інший варіант реалізації архітектури БЗ має на увазі створення системно орієнтованої бази. Тобто вона створюється під конкретну тематичну систему, що використовує краулер у якості пошуковця збирача, з подальшою обробкою отриманої інформації, та її розподілу по відповідним сутностям БЗ.

Незалежно від реалізації БЗ повинна вміщати частину, що відповідає за тематичний розподіл інформації, зібраної зі сторінок Інтернету.

Як видно з рис. 1, структурно, БЗ можливо розділити на 3 частини. Перша, включає в себе інформацію, що надходить з краулера, тобто індекси, за якими було оцінено тематичну інформацію та відповідно оброблений масив даних зі сторінок знайденого ресурсу. Потім ця інформація, є базовою(вхідною) для іншої підсистеми – машини пошуковця. Всі дані, що раніше були знайдені та оброблені краулером формують відповідь на запит пошукової системи за обраним правилом мулевої логіки. Друга частина є так званим представленням метаданих. Тобто тут зберігаються довідники загального призначення, та перелік ресурсів Інтернет простору, що внесені до списку задач краулера. Ця частина використовується як краулером, так і пошуковою машиною. Третя частина, є не обов’язковою і залежить від кінцевої прикладної системи що використовує пошукового робота. Ця частина є описовою для структури систем прикладного призначення, і при конструюванні загальної системи може бути виділена в окрему базу знань або об’єднана з існуючою.

Структурне та функціональне поєднання цих трьох модулів в кінцевому випадку дає нам повнофункціональну систему пошуку цільового призначення.

Програмне забезпечення

Програмне забезпезпечення представляє собою опис структурної(прогрманої) частини підсистем, технологій що були використані при створені пошукової системи. Опис цих частин також доцільно привести з точку зору кожного модуля.

Модуль пошуку тематичної інформації

Загалом кожен з варіантів реалізації робота-пошуковця можливий з використанням стандартних інструментів та засобів багатьох об’єктно орієнтованих мов програмування. Так наприклад використовуючи мову C# платформи .Net Framework [5], ми можемо створити асинхронний вебзапит до обраного ресурсу і отримати повні дані з сторінки з отриманої веб-відповіді:

```
HttpWebRequest request = (HttpWebRequest)
     HttpWebRequest.Create(requestData.BaseRequestURL);
request.AllowWriteStreamBuffering = true;
request.AllowAutoRedirect = requestData.AllowAutoRedirect;
request.Method = requestData.Method;
request.ContentType = requestData.ContentType;
request.UserAgent = requestData.UserAgent;
```

 ISSN 1562-9945

```
request.Accept = requestData.Accept;
request.CookieContainer = requestData.Cookies;
if (requestData.WriteContentLength > 0) {
  request.ContentLength = requestData.WriteContentLength;
  request.GetRequestStream().Write
   (requestData.WriteContentEncoded,0,
    requestData.WriteContentLength);}
if (requestData.Header.Count > 0) {
  foreach (KeyValuePair<string, string>header
                                  in requestData.Header){
    request.Headers.Add(header.Key, header.Value);}}
if (!string.IsNullOrEmpty(requestData.ProxyServer)){
  WebProxy proxy = new WebProxy();}
NetworkCredential credent =
      new NetworkCredential(requestData.ProxyUserName,
                        requestData.ProxyUserPassword);
proxy.Credentials = credent;
proxy.Address = new Uri(requestData.ProxyServer);
request.Proxy = proxy;
AsyncCallback async = new AsyncCallback(SiteResponse);
IAsyncResult result =
             request.BeginGetResponse(async, request);
ThreadPool.RegisterWaitForSingleObject
  (result.AsyncWaitHandle,
   new WaitOrTimerCallback(TimeoutCallback),
   request, DefaultTimeout, true);
```

Тут *SiteResponse* – функція, котрій буде передано відповідь на наш веб-запит. Отже циклічно використовуючи даний код, ми зможемо перебрати переданий нам набір веб-адрес в паралельних потоках, дані з котрих будуть надходити до вихідного модулю, що через спільний інтерфейс доступу передаються до робота-аналізатора.

Модуль аналізу та збору інформації

При описі модуля аналізу та збору інформації, нас більше цікавить, як саме наша система буде отримувати необхідні данні зі сторінки. Тут на допомогу приходять декілька інноваційних технологій в сфері Web та XML [6] запитів.

Мова XPath основана на представленні XML документа у вигляді дерева, і надає можливість навігації всередині дерева, вибирати вузли за різними критеріями. В просторіччі (хоча це не офіційна специфікація) XPath вираз часто називають просто XPath.

На початку вмотивований бажанням надати загальний синтаксис і модель поведінки між XPointer та XSLT, XPath швидко здобув визнання розробників як мова малих запитів, і його підмножини використовуються в інших специфікаціях W3C, як наприклад XML Schema та XForms.

Однією з головних ідей специфікації XPath, це ідентифікувати набір вузлів, які задовольняють специфічній умові для XML документа. По суті справи, дані отриманні зі сторінки нашим пошуковим роботом, представлені у вигляді HTML документу. Тобто чітко структурованого документа з набором деревоподібних тегів з атрибутами. Маючи спеціальні засоби ми можемо легко перевірити вірність синтаксису такого документа та перевести його до відповідного структурованого XML документу. А далі за справу вже береться XPath та XSH. Ідея полягає в тому, що завдяки спеціально структурованим запитам (схожих на SQL запити), ми можемо легко дістати будь які дані зі створеного дерева XML документу. Для прикладу можна привести невеликий код типової сторінки що має певну текстову інформацію з певним характерним стильовим обрамленням:

<TR ...><TH ...>
<A NAME="text"...>
< TH...>< TR...>

І це все закінчується наступним аналогічним рядком

<TR ...><TH ...>
<A ...>
< TH...>< TR...>

В даному випадку “. . . ” використано для вказання додаткової, менш важливої інформації. Задача стоїть в тому щоб отримати текст що знаходиться в тезі <A NAME="text"...>.

Отже, тепер, коли перед нами поставлено конкретне завдання, в нас є можливість використати два різні підходи, один використання xsh, а інший використання HTML::TreeBuilder::XPath. (Можна побачити, що код майже ідентичний, тому доцільно привести тільки один приклад.).

Використовуючи діалоговий режим xsh команди, можна знайти вирази XPath для даних, в яких ми зацікавлені. Специфікація для вузла що включає потрібну нам інформацію виглядатиме наступним чином: *//tr[th/a/@name ='text']*

Треба відмітити, що ми не піклуємося, пишучи це за допомогою XPath про будь-які інші властивості, або, навіть, якщо наше значення *text* знаходиться в одиничних або подвійних лапках (якщо цитується взагалі). Це значно краще, ніж, використання регулярних виразів.

Що насправді означає цей вираз? Спочатку, буде знайдемо всі вузли з ім’ям *tr*, після чого в даному вузлі будуть шукатись всі інші вузли на одному рівні, що мають ім’я *th*. Далі для знайденого вузла буде шукатись вузол з ім’ям a, та той, що маю атрибут *name* зі значення *text*. Прохід по вузлам починаючи з *th* буде відбуватись циклічно.

Формування набору таких запитів, до знайденої сторінки, дозволяє отримати будь-які данні з документу, незважаючи на коректність чи синтаксис використаний при створенні HTML документу. Ще одною перевагою використання такого підходу є те, що технологія XPath дозволяє формувати запити з використанням не латиномовних символів, тож для

обробки документу можна використовувати атрибути тегів, що вміщають текст кирилиці.

Отриманні данні після обробки документу модулем аналізу та збору інформації можна легко привести до єдиного стандарту, що згодом можна занести до бази знань.

Модуль синхронізації даних з базою знань

Для оптимізації створення БЗ [3] та її зв'язку з фізичними сутностями існуючої тої, чи іншої системи, можна використати доволі новий засіб NHibernate, що надає більш спрощену структуру взаємодії.

Метою NHibernate є звільнення розробника від значних типових завдань із програмування взаємодії з базою даних. Розробник може використовувати NHibernate як при розробці з нуля, так і для вже існуючої бази даних.

NHibernate піклується про зв'язок класів з таблицями бази даних (і типів даних мови програмування із типами даних SQL), і надає засоби автоматичної побудови SQL запитів й зчитування/запису даних, і може значно зменшити час розробки, який зазвичай витрачається на ручне написання типового SQL і JDBC коду. NHibernate генерує SQL виклики і звільняє розробника від ручної обробки результуючого набору даних, конвертації об'єктів і забезпечення сумісності із різними базами даних.

Основою використання сутностей NHibernate, є створення так би мовити файлів мапінгу. Кожен такий файл, повністю описую яке поле таблиць БЗ відповідає фізичній сутності системи(классу). Перевагами проектування систем з використанням засобів NHibernate, на відмінну, від стандартних бібліотек доступу до баз даних об'єктно орієнтованих мов програмування, є спрощена система доступу до даних. Розробник може легко модифікувати класи системи під структури БЗ, не хвилюючись про зміну всіх запитів, адже, за формування відповідних запитів на оновлення та вибірку даних безпосередньо з БЗ, відповідає відповідно система NHibernate. Також така структура менш схильна до використання помилкових запитів, та втрати інформації при розривів транзакцій. Таким чином вся увага розробників зводиться до розробки відповідних, необхідних, фізичних сутностей системи та їх віднесення до відповідних полів БД.

Недоліки такого конструювання, зв'язку моделі даних з даними бази, невеликі, та все ж присутні, і пов'язані вони більше з накладеними обмеженнями на можливості NHibernate. Нажаль цей засіб дозволяє пов'язувати лише одну сутність моделі системи з однією сутністю бази даних. Тож створення більш складних взаємопов'язаних типів поки що неможливо. Проте кожен складний тип можна розкласти на прості і пов'язати їх між собою.

Прикладом структури типової сутності з використанням технології NHibernate може бути приблизно таким.

<?xml version="1.0" encoding="utf-8" ?>

```
<hibernate-mapping xmlns="urn: nhibernate-mapping-2.2" namespace="NHibernate_1" assembly="NHibernate_1">
<class name="Urls" table="Urls">
<id name="ID" unsaved-value="0">
<column name="ID" not-null="true" />
 <generator class="identity"/>
 </id>
 <property name="Url" />
<property name="PageName" />
 </class>
</hibernate-mapping>
```

Даний код описує файл мапінгу, для співвідношення фізичного класу системи *Urls* з таблицею бази даних *Urls.*

<class name="Urls" table="Urls"> - описує клас що буде представляти обрану таблицю з бази даних.

Тег *id* – описує унікальний ідентифікатор, за допомогою класу *identity.*

<property name="Url" /> та <property name="PageName" /> - описуєть інші свойства нашого класу.

Таким чином створивши такий клас мапінгу та відповідний файл конфігурації з’єднання NHibernate з базою даних, ми отримаємо клас, що здатен без додаткового коду запитів, взаємодіяти з базою даних, тобто оновлювати, видаляти, додавати та отримувати данні.

Висновок

Побудова сучасного робота-краулера є багатокроковою та свого роду раціоналізаторською роботою. Не існує єдиного стандарту, що описує структуру таких систем. Як правило розроблюється система галузевого призначення, під котру згодом пишеться, більш вузького профілю, краулер. Винятками, мабуть, що становлять найбільші сучасні пошукові системи Інтернету, проте і вони більше спеціалізуються на пошуку та індексації інформації, а не на її збиранні та аналізі.

Описаний в цій статті алгоритм побудови такого краулера, дозволяє спроектувати його таким чином, що він міг легко інтегруватись, до будь-якої тематично-галузевої системи, без значних модифікацій. Створення такого робота з відкритим узагальненим інтерфейсом взаємодії, надає можливість розробникам вільно користуватись всіма перевагами сучасних WEB та XML засобів. Введення додаткових інноваційних систем взаємодії та обробки даних, дозволяю в декілька раз підвищити якість інформації та швидкість її обробки, на відміну від стандартних систем що їх пропонують розробникам.

Використання запропонованого підходу, також дозволяє використовувати отриманий робот для більш потужних засобів, при менш сприятливих умов. Тобто модифікував його певні частини та вдосконаливши дещо алгоритм пошуку та проходження по сторінкам, ми можемо отримати результати, схожі на те нібито, всю роботу по пошуку і зчитуванню

даних виконувала людина зі звичайного браузера. Тим самим ми захищаємо нашу систему від робо-блокераторів, що їх інколи встановлюють на ресурсах веб-простору Інтернету.

Література

1. Веб-аналитика: анализ информации о посетителях веб-сайтов. Web-аналитика, Авинаш Кошик, 464 стр., с ил.; ISBN 978-5-8459-1480-4, 978-0-470-13065-0; формат 70x100/16; мягкий переплет; CD-ROM; 2008, 4 кв.; Диалектика.
2. Поисковые системы и продвижение сайтов в Интернете; Колисниченко Денис Николаевич, 272 стр., с ил.; ISBN 978-5-8459-1269-5; формат 70x100/16; мягкий переплет; серия Краткое руководство; 2007, 3 кв.; Диалектика.
3. Базы знаний интеллектуальных систем, Гаврилова Т.А., Хорошевский В.Ф., 1-е издание, 2001 год, 384 стр., формат 17x24 см, твердая обложка, ISBN 5-94723-449-1; Питер.
4. Проблемы семантического анализа лексики. Изд.2, Шмелев Д.Н., Переплёт: мягкий; Объём: 280 стр.; ISBN: 9785382006161; Формат: 60x90/16; 2008 г.;Издательство ЛКИ.
5. Основы разработки приложений на платформе Microsoft .NET Framework, Нортрап Тони, Вилдермьюс Шон, Райан Билл; Издательство Питер, Русская Редакция; 2007 г.; 864 с.; ISBN: 978-5-469-01659-5, 978-5-7502-0297-3, 978-0-7356-2277-7; Язык русский.
6. XML. Справочник, Эллиот Расти Гарольд, В. Скотт Минс; Издательство Символ-Плюс; 2002 г.; 576 с.; ISBN: 5-93286-025-1; Язык руський

Получено 04.11.2008