УДК 681.3.06

Романенко В. Г.

ЗАСОБИ ПРОЕКТУВАННЯ ВЕБ-ІНТЕРФЕЙСІВ РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Статтю присвячено розробці ефективного підходу до реалізації веб-інтерфейсів розподілених інформаційних систем на платформі Windows. Запропоновано метод реалізації веб-інтерфейсу на базі процесора шаблонів, який дає змогу розділити представлення, дані та код. Такий підхід апробовано в реалізованій автором системі керування подіями.

Вимоги сучасного бізнесу та виробництва викликають необхідність створення інформаційних систем на підприємствах різного рівня - від великих корпорацій до маленьких фірм. І чим більший масштаб підприємства, тим більшою є потреба в мережних системах, які дають змогу одночасно великій кількості користувачів мати доступ до ресурсів системи. Зручним варіантом реалізації інтерфейсу таких розподілених систем останнім часом стало середовище всесвітньої павутини (веб). Воно дозволяє при мінімальних вимогах до клієнтських робочих місць (необхідний лише веб-браузер) організувати доступ до широкої функціональності інформаційної системи.

У цій роботі розглядається проблема побудови веб-інтерфейсів розподілених інформаційних систем. Метою є вироблення ефективного підходу до реалізації веб-інтерфейсу на платформі Microsoft Windows для розподілених систем, що побудовані на базі трирівневої та багаторівневої архітектури. У таких системах за генерацію веб-інтерфейсу відповідає презентаційний рівень, який реалізується як набір скриптових кодів (скриптів). ASP, або Active Server Pages,- це технологія Microsoft, яка дає змогу виконувати серверні скрипти на веб-сервері [1].

Основними критеріями ефективності було обрано:

1) швидкість розробки системи;
2) швидкодію спроектованого програмного коду;
3) час відгуку системи на запит користувача.

Одним з важливих критеріїв є швидкість розробки системи, оскільки в сучасних умовах необхідно мати можливість швидко створювати інформаційні системи та адаптувати їх до мінливих вимог бізнесу. Як засіб досягнення цієї мети було вироблено підхід, який дозволяє одночасно і незалежно працювати над розробкою системи дизайнерам, програмістам бізнес-логіки та веб-програмістам. Цей підхід містить у своїй основі спроектовану і реалізовану систему процесорної обробки шаблонів веб-сторінок, яка ефективно розділяє дизайн, дані бізнес-об'єктів та ASP-код.

Цей підхід було застосовано в розробці системи керування заходами, що використовується всесвітньою мережею FirstTuesday (http://www.firsttuesday.com).

Не менш важливими критеріями є швидкодія та час відгуку системи на запит користувача, оскільки система керування заходами зазнає підвищеного навантаження в дні проведення заходів. Важливо, щоб система була здатна обробити збільшену кількість користувачів у ці дні та щоб показники її роботи при цьому не погіршилися.

Windows DNA як платформа розподілених веб-систем

Платформою для реалізації розподіленої системи керування подіями було обрано Windows DNA. При сучасному рівні розвитку програмних систем від різних постачальників вони часто відповідають одна одній за своєю функціональністю, і тому вибір залежить більше від другорядних факторів - досвід розробників, економічна доцільність обрання тієї чи іншої платформи, наявність пропозицій провайдерів, які пропонують програмне забезпечення певного постачальника. Обрання платформи для реалізації цієї системи було зумовлено в першу чергу досвідом розробки попередніх версій системи на платформі від фірми Microsoft та умовами партнерства з провайдерами, які пропонували рішення саме на базі цієї платформи. Тому зараз ми детальніше розглянемо особливості розробки на платформі фірми Microsoft, яка має назву Windows DNA.

Windows DNA - це архітектура розподілених Інтернет-застосувань на платформі Windows

(Windows Distributed interNet Applications Architecture) [3]. Вона включає в себе операційну систему Windows NT або Windows 2000 та ряд супутніх продуктів. Інфраструктура, що реалізує проміжний рівень розподіленої системи, у випадку Microsoft включає веб-сервер Internet Information Server (IIS) та сервер транзакций Microsoft Transaction Server (MTS). Усе це базується на технологіях Distributed Component Object Model (COM/DCOM), а також використовує можливості операційної системи Windows NT/2000. Важливим моментом є те, що Microsoft Transaction Server (так само як і Internet Information Server) не вимагає ліцензування. А Windows 2000 включає їх до складу операційної системи. Вузьким місцем MTS є служба імен Windows NT. І вирішення цієї проблеми прямо пов'язано з виходом Windows 2000 з його службою Active-Directory. Маються на увазі обмеження для Інтранет, тому що в Інтернет-застосуваннях рятує IIS.

Як висновок з вищесказаного слід зазначити, що Windows DNA представляє розвинену архітектуру для розробки складних розподілених систем. Вона має широкі можливості та задовольняє вимоги інтероперабельності з іншими середовищами. її орієнтація на веб-застосування надає широкий спектр можливостей розробникам і дає змогу будувати повнофункціональні системи на базі веб-технологій.

Програмна архітектура системи

Практичну частину роботи присвячено розробці та аналізу системи керування заходами. Конкретна система, над якою було проведено роботу, використовується всесвітньою мережею FirstTuesday (http://www.firsttuesday.com), яка поєднує 50 міст у 30 країнах світу. У кожному місті є окремий осередок, що займається організацією і проведенням заходів, присвячених висвітленню новітніх технологій у галузі Інтернет та інших галузях IT-індустрії та телекомунікацій [2].

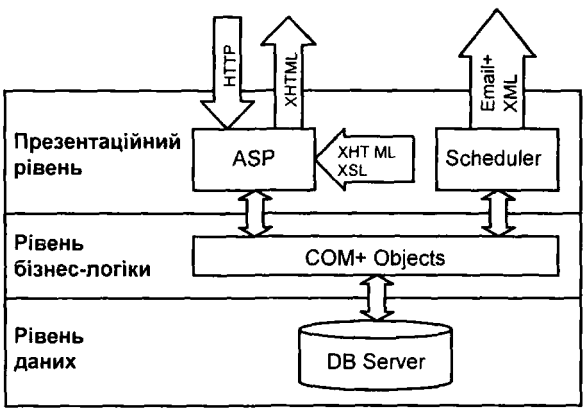

Рис. 1. Розподілена архітектура системи

Цій системі притаманні певні особливості, які мали вплив на вибір її архітектури.

Глобальний характер системи. Більшість подібних систем є локальними чи функціонують у межах корпорації всередині окремої мережі чи мереж. Специфіка даної системи в її розповсюдженості по всій земній кулі.

Особливі вимоги до продуктивності системи. Сама назва мережі FirstTuesday відбиває ту особливість, що основні заходи проводяться по перших вівторках кожного місяця. Це приводить до підвищеної активності користувачів у цей період та зростаючого навантаження на систему. Такі високі пікові навантаження вимагають спеціальних методів підвищення продуктивності.

Розглянемо архітектуру розподіленої системи керування заходами.

Система керування заходами складається з типових для розподілених систем компонентів (рівнів). Зробимо короткий огляд технологій, використаних для реалізації кожного рівня:

1. **Презентаційний рівень (Presentation Layer).** Презентаційний рівень складається з ASP-сторінок з широким використанням XML та XSL.
2. **Рівень бізнес-логіки (Business Layer).** Цей рівень складається з набору COM+ об'єктів.
3. **Рівень даних (Data Layer).** На цьому рівні систему представлено набором збережених процедур, що виконуються на SQL-сервері.

У подальшому дослідженні нас цікавитиме презентаційний рівень системи, а також його зв'язки з рівнем бізнес-логіки. Традиційні підходи до розробки веб-застосувань, такі як CGI та ISAPI [4], не задовольняють потреби до розробки веб-інтерфейсів великих інформаційних систем з огляду на їх низьку продуктивність (CGI) чи складність у розробці (1SAPI). Ефективною є комбінація скриптових мов для реалізації презентацій ного рівня та компонентного підходу для реалізації бізнес-рівня (наприклад, ASP+COM чи JSP+JavaBeans).

Але використання скриптових мов для побудови веб-інтерфейсу, а простіше кажучи - вебсторінок, легко може привести до суміщення коду презентаційного рівня та форматування веб-сторінки. Результатом цього є складність підтримки системи, оскільки людина, що займається дизайном, має втручатися в код програми, а з іншого боку, створює складності для відображення в html даних, отриманих від бізнес-об'єктів.

Для вирішення цієї проблеми було розроблено механізм шаблонів, який дав змогу відокремити презентаційний код (ASP), дизайн веб-сторінок та форматування даних бізнес-об'єктів.

Процесор шаблонів

Розглянемо детальніше архітектуру презентаційного рівня, центральною частиною якої є процесор шаблонів.

Розглянемо функціонування цієї архітектури. Клієнт, який являє собою веб-браузер, генерує запит до системи. Цей запит обробляється певною ASP-сторінкою. Вона приймає вхідні дані від клієнта у вигляді полів HTML форми чи елементів рядка запиту (query string). Ці дані обробляються та визначається шаблон HTML-сторінки, яка має бути відображена користувачеві. Під час виконання такої підготовчої обробки код сторінки може звертатися до COM-об'єктів системи для отримання необхідних даних, а також виконання певних дій, ініційованих користувачем.

Наступним кроком є передача сформованих даних до процесора шаблонів. Процесор завантажує відповідний шаблон та обробляє його. У чому полягає процес такої обробки? По-перше, шаблон може містити поля форм та інші елементи сторінки, які мають бути динамічно заповнені на підставі даних, переданих користувачем та отриманих від COM-об'єктів на попередньому етапі. Ці дані готуються кодом ASP-сторінки та передаються процесору як параметри його виклику. Крім того, певні частини сторінки динамічно генеруються на основі даних, отриманих процесором безпосередньо від COM-об'єктів.

Наприклад, при редагуванні певного об'єкта системи користувач змінює на формі його назву. При відсиланні на сервер цих даних ASP-код визначає, що має бути виконане оновлення даних відповідного об'єкта. Для цього викликається екземпляр цього об'єкта, який і виконує це завдання (та інкапсулює в собі бізнес-логіку цього процесу). Потім від нього отримуються оновлені дані, які передаються процесору. Той вставляє ці дані у форму, яка має бути повернута користувачеві. Крім того, ця форма містить певний динамічний контент, наприклад список інших об'єктів. Цей список процесор отримує безпосередньо від бізнес-об'єкта. Дані передаються у вигляді XML. Далі вони форматуються з допомогою XSL-перетворення та вставляються в сторінку. При цьому копія цих даних зберігається в кеші, якщо це має сенс (такі дані рідко змінні, тому їх можна кешувати). Таким чином ця схема дає змогу цілком відділити дані та їх представлення. Для зміни представлення досить відредагувати відповідні XHTML-шаблони та XSL-стилі, змінювати код для цього не потрібно.

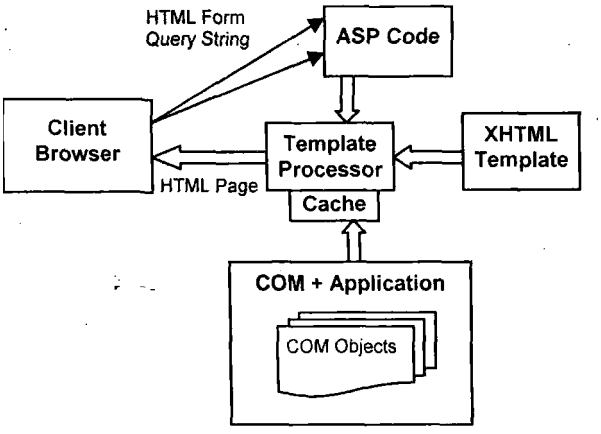

Рис. 2. Архітектура реалізації презентаційного рівня системи

Аналіз продуктивності системи та можливості покращення її ефективності

У ході практичної роботи над реалізацією системи було створено першу її версію. Під час випробувальної експлуатації виявилося, що система хоч і задовольняє багато вимог **(адаптивність, доступність** та **масштабованість),** проте має певні недоліки в ефективності роботи. А саме - час відгуку системи був завеликий навіть при незначних її завантаженнях. Тому було проведено ретельний аналіз, який мав допомогти знайти можливості поліпшення ефективності роботи системи. Для аналізу завантаженості серверів було використано утиліту Microsoft Performance Monitor, яка постачається у складі Microsoft Windows 2000 Server.

Таблиця 1. **Показники ефективності роботи системи**

	SQL	Web1	Web2
Апаратна конфігурація	Dual 550 MHz CPU 512 Mb RAM	550 MHz CPU 256 Mb RAM	
Операційна система	Windows 2000 Server	Windows 2000 Advanced Server	
Черга запитів сторінок	X	10	2
Відмови в обслуговуванні запитів	X	0	0
Кількість запитів за сек. (макс.)	X	42	24
Максимальне завантаження процесора	55 %	34 %	21 %
Середнє завантаження процесора	5 %	5 %	3 %
Максимальна довжина дискової черги	2	2	1
Довжина черги потоків сервера	0	0	0

Виходячи з цих показників можна зробити такі висновки. Завдяки балансуванню навантаження двох веб-серверів їх пікове завантаження не перевищує 34 % і 21 % відповідно для першого і другого серверів. Пікове навантаження система зазнає під час проведення чергових заходів (кожного першого вівторка місяця). У ці дні кількість звертань до серверів є максимальною (34 та 21 відповідно). Що ж стосується сервера бази даних, його пікове завантаження (55 %) припадає на розсилання масових повідомлень напередодні чергового заходу. Оскільки такі дії виконуються вночі під час найменшого завантаження системи, це не погіршує її працездатність. У цілому показники продуктивності системи вказують на те, що вона ефективно справляється з піковими навантаженнями.

Окреме дослідження провадилось для з'ясування часу відгуку системи на запит користувача при різноманітних навантаженнях. Для цього використовувалась утиліта Microsoft Web Application Stress [8]. З її допомогою було заміряно середню кількість згенерованих ASP-сторінок за секунду. Частина сторінок викликала об'єкти прямо, а інша частина використовувала шаблонний процесор.

Таблиця 2. **Час відгуку системи при різних реалізаціях**

Реалізація	Кількість користувачів			
	10	**100**	**500**	**1000**
ASP/COM	800	780	770	760
ASP/Template Processor	500	490	485	480

Отже, можна зробити очевидний висновок, що показники швидкодії спроектованої системи знизилися майже вдвічі після застосування архітектури, що базується на шаблонному процесорі. Застосування процесора шаблонів є дуже важливим кроком у проектуванні та реалізації всього презентаційного рівня, оскільки воно дало змогу дуже ефективно розділити управляючий ASP-код, дані та представлення (інтерфейс користувача). Над цими різними частинами надалі працюють різні люди: дизайнери та верстальники займаються HTML-інтерфейсом, програмісти - програмним кодом.

Результати тестування свідчать також про те, що швидкодія є вузьким місцем системи. Такі результати не є несподіваними й досить легко пояснюються. По-перше, реалізація такої складної системної компоненти мовою Visual Basic не є достатньо ефективною внаслідок високорівневої природи самої мови. Visual Basic орієнтована на швидку розробку компонент бізнес-логіки, а не написання ефективного коду. Звичайно для пілотної реалізації її можливостей було достатньо, а швидкість розробки була основним фактором. Але для реальної системи з великими навантаженнями та високими вимогами щодо ефективності таку критичну компоненту було вирішено реалізувати на C++.

Другим виявленим у результаті аналізу аспектом, що впливає на ефективність роботи процесора, було використання бібліотек для парсінгу XML-документів. Парсер фірми Microsoft MSXML 3.0, що використовується в системі, базується на моделі DOM [9]. При розборі документу він будує дерево його структури в пам'яті. Звичайно, це досить зручно для маніпуляцій над документом, для його обробки, але вимагає великих ресурсів (особливо зважаючи на великий обсяг деяких XML-документів, які доводиться обробляти процесору). Крім того, побудова дерева і маніпуляція ним є дуже повільною і неефективною. Тому було прийнято рішення для реалізації наступної версії використовувати потокові парсери, які відповідають відносно новій специфікації **API** SAX (Simple API for XML) [10]. Цей API реалізовано в останній версії парсера фірми Microsoft MSXML 4.0. Потокова та базована на подіях обробка XML-документів, яка реалізується в цій специфікації, якнайкраще відповідає моделі обробки шаблона процесором та є набагато ефективнішою, ніж DOM API. Проте слід зауважити, що DOM API все ще буде використовуватися для виконання XSL-перетворень даних, отриманих від бізнес-об'єктів.

Висновки

Розроблений підхід до реалізації веб-інтерфейсів розподілених інформаційних систем на базі процесора шаблонів прискорив розробку систем. Він також підвищив ефективність роботи розробників через розділення представлення, даних та коду. В результаті дослідження було виявлено вузькі місця в реалізації архітектури. Перспективою розвитку даної роботи є оптимізація критичних частин системи, а саме процесора шаблонів, з використанням більш ефективних мов програмування та бібліотек парсінгу XML.

1. *ГомерА.. Суссман Д.. Френсис Б.* Active Server Pages 3.0 для профессионалов. В 2-х т.- М.: Лори, 2002.
2. *Ягофаров Т.* UNEX включается в международную сеть First Tuesday *II* Компьютерное обозрение- 2001- № 6 (http://itc.ua/article.phtml?ID=5172&IDf=2704).
3. Professional Windows DNA.- Wrox Press Inc., 2000.
4. *Фролов А. В., Фролов Г. В,* Создание Web-приложений: Практическое руководство- М.: Русская Редакция, 2001.
5. *Kirk S.* Duwamish Online Application Architecture- 2001.- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnduwon/html/d5ntierarch.asp

Leake G. Architecture Decisions for Dynamic Web Applications: Performance, Scalability, and Reliability- 2000.- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnile/html/docu2kbench.asp.

Strahl R. Scaling Web Applications with Windows 2000 Advanced Server's Network Load Balancing- West Wind Technologies (http://www.west-wind.com/presentations/loadbalancing/loadbalancing.htm).

MS Web Application Stress Tutorial.- Microsoft (http://www.microsoft.com/technet/itsolutions/intranet/downloads/webtutor.asp?frame=true).

World Wide Web Consortium. W3C Document Object Model (http://www.w3 .org/DOM/).

10. Simple API for XML project (http://www.saxproject.org/).

V. G. Romanenko

AN APPROACH TO BUILDING WEB-INTERFACES OF DISTRIBUTED INFORMATION SYSTEMS

The article touches upon the problem of effective development of web-interfaces of distributed information systems on Windows platform. An approach is suggested for development based on templates processor, which enables effective separation of view, data and code. This approach is applied by author to development of event management system.