після серіалізації утвориться множина $\{<id_1, S(A_1)>, ... <id_{An}, S(A_n)>\}$, де $S(A) = < V(A), <id_1, ... id_{Bn}> >$, а $B_1, ... B_n$ – об'єкти, посилання на які безпосередньо міститься в об'єкті A. У програмі роль ідентифікатора об'єкта виконує його адреса, але замість неї зручніше обрати деякі ідентифікатори у процедурі серіалізації для більш легкого читання людиною отриманого образу. Під час серіалізації потрібно включити список адрес уже записаних об'єктів як для ведення списку ідентифікаторів, так і для виявлення можливих циклів у разі обходу графу методом у глибину.

Слід зазначити, що результат серіалізації дерева легко подати у вигляді XML, коли вміст кожного об'єкта є одним елементом з деяким набором атрибутів і вкладених елементів, тому, розглядаючи проблему серіалізації, можна сформулювати рекомендацію, щоб класи, передані між віддаленими компонентами, були коренем дерева об'єктів. Зокрема це дерево може бути навіть виродженим, тобто клас не містить полів-посилань взагалі.

2.8. Базові технології подання інформації в розподілених системах

2.8.1. Вимоги до прикладних програм серверної сторони

Розглядаючи платформи для створення прикладних програм серверної сторони, необхідно виокремити такі основні підходи: безпосередня обробка запитів і формування відповідей; вбудовування програмного коду в шаблони HTML-сторінок.

Перший підхід надає найбільші можливості з керування обробкою і підвищенням продуктивності, оскільки він передбачає передачу всіх даних про запит безпосередньо виконуваного коду, який може як сформувати відповідь зі сторінкою для користувача, так і відкрити процес передачі потоку бітів, наприклад для передачі зображення. Однак за такого підходу всі дані для передачі формуються програмно, що уповільнює розробку простих сторінок і ускладнює взаємодію між розробником дизайну сторінки і програмістом. Прикладами цього підходу є технології CGI (Common Gateway Interface), Java Servlets.

Другий підхід використовує шаблони сторінок користувача, оформлені таким чином, щоб дозволити вставляти в них ділянки програмного коду. Цей підхід особливо ефективний під час створення простих прикладних програм, основна інформація в яких статична, а динамічна інформація може бути генерована простими програмними конструкціями. У процесі розробки складних програмних систем цей варіант ускладнює взаємодію між компонентами і реалізацію складної архітектури, а також він менш ефективний за продуктивністю й обмежує можливості з реалізації складних сторінок. Прикладами цього підходу є найпоширеніші нині технології Personal Home Page (PHP), Active Server Pages (ASP), Java Server Pages (JSP).

Крім різних підходів до генерації сторінок сучасні платформи розробки складних Web-систем мають задовольняти вимогам, дотримання яких робить систему зручною у використанні:

- платформна незалежність;
- мова реалізації;
- продуктивність;
- масштабованість;
- можливість розширення й інтеграції;
- простота використання, наявність засобів розробки;
- наявність необхідних програмних бібліотек.

Розглянемо найбільш популярні нині платформи, їх особливості, а також оцінку з погляду наведених критеріїв.

2.8.2. Огляд базових технологій

Нині є безліч розроблених технологій серверної сторони, як комерційних, так і вільно поширюваних.

Платформи розглядаємо з погляду побудови на них складних гетерогенних Web-систем, тому деякі з популярних технологій не наводимо в детальному огляді через неможливість або недоцільність їх використання як базової платформи. Наприклад, технологія ISAPI та інші технології, які розширюють можливості Web-серверів, не підходять для застосування внаслідок прив’язки

до конкретного Web-сервера. Наведемо тільки основні технології, потенційно здатні створити складні гетерогенні Web-системи.

Технологія Common Gateway Interface. Технологія CGI вирізняється серед інших розглянутих технологій тим, що є найбільш низькорівневим стандартом інтерфейсу, який забезпечує зв'язок зовнішньої програми з web-сервером.

Сам протокол розроблено таким чином, щоб можна було використовувати будь-яку мову програмування та працювати зі стандартними пристроями введення/виведення. Оскільки така можливість надається на рівні операційної системи, то використовують або скрипт, написаний на відповідною мовою програмування, або простіше його подання - командний файл.

Розглянемо основні переваги та недоліки технології CGI за окремими критеріями:

– технологія CGI не передбачає особливих обмежень для використання платформи та web-сервера, тому працює на всіх популярних платформах і web-серверах, а також технологія не прив'язана до конкретної мови програмування й може бути використана будь-якою мовою, що працює зі стандартними потоками введення/виведення;

– продуктивність CGI-програм невисока, основною причиною чого є те, що у разі чергового звертання до сервера для роботи CGI-програми створюється окремий процес, який вимагає великої кількості системних ресурсів;

– технологія не передбачає вбудованих засобів масштабованості, це потребує додаткового доопрацювання програмного забезпечення розробниками;

– CGI-програма являє собою готовий до виконання файл, що перешкоджає легкому розширенню системи.

Із зазначених причин на сьогодні віддають перевагу більш розвиненим платформам, які надають більше зручності розробникам та мають підвищену продуктивність. Однак велика кількість уже розробленого програмного забезпечення змушує розвивати й використовувати технологію CGI, потрібну для розуміння принципів роботи високорівневих платформ.

Технологія Personal Home Page. Технологія PHP набула значного поширення завдяки своїй безкоштовності й підтримання найбільш популярних платформ, оскільки грунтується на принципі побудови сторінок із шаблонів, який вперше запропоновано в технології ASP. Технологія PHP розвиває і доповнює цей принцип. Сторінки PHP мають вигляд звичайних HTML-сторінок, у яких можуть використовуватися спеціальні теги *<? Php i?>*, між якими вставляються рядки програмного коду спеціальною мовою сценаріїв PHP.

Принцип шаблонів дозволив розробникам писати програми набагато швидше і без помилок, властивих традиційним CGI-програмам, які пересилають HTML-вміст у потік виведення. На сьогодні низка систем, побудованих на шаблонах, містить як прості сторінки з вибірками з бази даних, так і великі прикладні програми електронної комерції, в основі яких лежить мова XML. Системи з використанням шаблонів мають велику популярність, оскільки найбільше підходять для типових сайтів. Такі рішення розробляються за допомогою технологій ColdFusion, PHP, JSP і ASP, з яких PHP є найбільш поширеною.

Розглянемо основні переваги та недоліки платформи:

– застосовувана в PHP мова проста і зручна, однак не є, в повному розумінні, об’єктно-орієнтованою;

– для PHP наявні великі бібліотеки, а також безліч вбудованих функцій для вирішення найрізноманітніших завдань;

– у разі використання PHP з Web-сервером Apache є можливість ефективного виконання ядра як розширення сервера, інакше продуктивність платформи невисока;

– власних засобів масштабування PHP не має, всі можливості з кластеризації цілком покладено на Web-сервер;

– можливості інтеграції обмежені наявністю лише функцій включення модулів і використання зовнішніх функцій, що не відповідає сучасним вимогам;

– підхід PHP, в основу якого покладено шаблони, за умови наявності значних переваг разом з тим має і серйозні недоліки.

Із загальних недоліків цього підходу, застосовуваних як до PHP, так і ASP, JSP необхідно звернути увагу на такі:

– файл-сторінку може підтримувати тільки людина, яка добре володіє як мовами програмування, так і мовою HTML, що вимагає підвищеної кваліфікації;

– один файл у конкретний момент часу може правити лише одна людина. Це означає, що працює або програміст, або дизайнер, тобто неможливий розподіл праці між групою фахівців;

– зберігання бізнес-логіки у файлах–сторінках у розподіленому між керівними елементами вигляді ускладнює її винесення в об’єкти другого рівня.

Таким чином, завдяки простоті використання, наявності великої кількості функцій і бібліотек, поширеності й підтримки більшості наявних Web-серверів і платформ, PHP є дуже зручним засобом розробки невеликих систем. У той же час, обмеження щодо продуктивності, масштабованості, неповноти як мови програмування, а також можливостей розширення та інтеграції перешкоджають використанню платформи для розробки масштабних систем.

Технологія Java Servlets. Технологію Java Servlets (сервлети) було розроблено компанією Sun Microsystems, щоб використовувати переваги платформи Java для розв’язання проблем технології CGI та API-розширень сервера, зокрема проблеми продуктивності, виконуючи всі запити як нитки в одному процесі. Сервлети також можуть легко розділяти ресурси і не залежать від платформи, оскільки виконуються всередині Java Virtual Machine (JVM).

Технологія має широкі функціональні можливості, оскільки велика кількість бібліотек надає найрізноманітніші засоби, необхідні під час розробки. Модель безпеки Java уможливлює точне керування рівнем доступу, наприклад дозволяючи доступ тільки до певної частини файлової системи. Завдяки обробці винятків Java-сервлети стають більш надійним засобом, ніж розширення серверів на мовах програмування C/C#.

Будь-який сервлет є класом Java, тому має бути виконаний усередині JVM **сервлет-контейнером** (servlet container, servlet engine), який завантажує клас сервлета під час першого звертання до нього або відразу в момент запуску сер-

вера за спеціальною вказівкою. Далі сервлет залишається завантаженим для обробки запитів, поки він не вивантажується явно, або до зупинення контейнера.

Технологія є поширеною і може бути використана для створення програмного забезпечення, яке працює з усіма популярними Web-серверами (Enterprise Server від Netscape, Microsoft Internet Information Server (IIS), Apache, Java Web-сервер від Sun).

Програмний інтерфейс дозволяє сервлетам обробляти запити на будь-якому рівні, у разі потреби використовуючи низькорівневі дані, такі як заголовки запитів, їх тип, що надає великої гнучкості під час розробки нестандартних обробників, наприклад для роботи з двійковим або мультимедійним вмістом.

Оскільки сервлети обробляють в одному процесі за рахунок створення потоків усередині, програмний код сервлетів, який міститься у потоці, має бути безпечним. Це накладає певну відповідальність на програміста, але за допомогою стандартних прийомів, таких як відмова від використання полів у класах сервлетів і зберігання необхідних даних у контексті або зовнішньому сховищі, такі властивості коду легко досягаються. За таких умов сервлети набувають таку перевагу, як масштабованість.

Отже, сервлети забезпечують компонентний, платформо-незалежний метод для побудови web-орієнтованого прикладного програмного забезпечення, яке не має обмежень продуктивності CGI-програм. Вони мають широкий діапазон доступних прикладних API, дозволяють використовувати всі переваги Java, легко розширюються і масштабуються, підтримуються всіма популярними Web-серверами. Все це робить їх засобом розробки великих Web-систем.

Технологія Java Server Pages. Технологія JSP від компанії Sun Microsystems стала надбудовою над технологією Java Servlets та забезпечує більш швидку і просту розробку web-прикладних програм за рахунок застосування підходу, який використовує шаблони програмування.

Для розуміння архітектури і переваг JSP необхідно знати технологію Java Servlets, оскільки вони тісно пов’язані. Сторінки JSP являють собою шаблони сторінок HTML, аналогічні шаблонам PHP і ASP. Основною відмінністю від інших подібних технологій є те, що код, який міститься всередині спеціальних тегів, не інтерпретується під час звертання до сторінки,

а попередньо компілюється в Java Servlet. Статичні ділянки шаблона перетворюються на виклики до функцій для їх розміщення в потік виведення. Код компілюється так, неначе він міститься всередині сервлета. Компіляція JSP сторінок у сервлети є трудомісткою, але виконується одноразово або під час першого звертання до сторінки, або у процесі запуску сервлет-контейнера.

Технологія JSP вдало поєднує підхід з використанням шаблонів до побудови сайтів і всі переваги Java-платформи, завдяки чому набула значного поширення як для створення професійних комерційних розробок, так і для відкритих безкоштовних проектів. Важливим кроком до розширення підходу з використанням шаблонів стали бібліотеки тегів (tag libraries), які створили можливість інтегрувати стандартні, сторонні або власні програмні компоненти у сторінки. Простота створення та використання зумовила популярність бібліотек тегів.

Завдяки роботі на основі Java технологія JSP не прив'язана до конкретної апаратної або програмної платформи, тому JSP є зручним рішенням для використання в гетерогенних середовищах.

Продуктивність технології обмежена такими об'єктивними особливостями архітектури: по-перше, сторінки мають бути відкомпільованими в сервлети, що потребує багато часу; по-друге, сервлети виконуються в середовищі Java, тобто в режимі інтерпретації.

Однак ці обмеження компенсуються додатковими можливостями, оскільки сучасні контейнери підтримують кластеризацію серверів, то навантаження перекладається на апаратне забезпечення. Це є економічно виправданим і простим рішенням. Завдання компіляції в сервлети є одноразовим та виконується або під час першого звертання, або у процесі запуску сервлет-контейнера, тому не позначається на загальній продуктивності системи у разі розгляду за достатній період.

Основними перевагами JSP є простота програмування, характерна для підходу з використанням шаблонів, наявність великої кількості сторонніх бібліотек, легкість їх застосування, потужні й різноманітні середовища розробки. Завдяки цим факторам JSP є найбільш перспективною базовою технологією розробки у процесі створення Web-сайтів. Однак у разі створення

складних Web-систем обмеження, які накладаються підходом з використанням шаблонів, стають серйозною перешкодою для розвитку цієї технології.

Технологія Microsoft.NET і середовище ASP.NET. Технологія .NET є новітньою розробкою компанії Microsoft і позиціонується як новий етап у розвитку засобів взаємодії між прикладними програмами. Нині вона доступна як доповнення .NET Framework до сімейства операційних систем Microsoft Windows, у продуктах Windows Server, а також ведуться роботи зі створення .NET Framework для інших операційних систем. Платформа .NET спрощує розробку прикладних програм і підвищує надійність коду, зокрема забезпечує автоматичне керування часом життя об'єктів, незалежні від мов програмування бібліотеки класів, обробку винятків і налагодження.

Основа .NET – середовище CLR (загальне середовище виконання мов) спирається на системні служби операційної системи і керує виконанням коду, написаного будь-якою сучасною мовою програмування. Набір базових класів надає доступ до сервісів платформи, які розробники можуть використовувати будь-якою мовою програмування. Середовище CLR і базові класи разом становлять основу .NET платформи, яка пропонує такі високорівневі сервіси:

– ADO.NET – нове покоління ADO, яке використовує XML і SOAP для обміну даними;

– ASP.NET – нова версія ASP, що дозволяє використовувати будь-яку (.NET сумісну) мову для програмування Web-сторінок;

– Windows Forms і Web Forms – набір класів для побудови користувацького інтерфейсу локальних та Web-орієнтованих прикладних програм.

Розгортання систем на платформі .NET здійснюється особливим чином; вихідні коди компілюються не в команди процесора x86 або інші машинні коди, замість цього компілятор створює код мовою проміжного шару програмного забезпечення, запропонованою Microsoft (Microsoft Intermediate Language – MSIL). Файл, що містить MSIL, може виконуватися на платформі будь-якого процесора, якщо операційна система має у складі .NET CLR.

Важливою складовою частиною платформи .NET є нове середовище ASP.NET. Можливості ASP.NET значно розширено, в його основі лежить інша платформа, і базовими мовами програмування для неї обрано C# і

Visual Basic, замість колишніх скриптових мов. У той же час, нова технологія дозволяє писати ASP-сторінки будь-якою відповідною мовою.

В ASP.NET закладено все для того, щоб зробити весь цикл розробки web-орієнтованих програм швидшим, а підтримку простішою. Основні можливості й принципи роботи ASP.NET такі:

- компілювання коду під час першого звертання;
- широкий вибір бібліотек компонентів, що постачаються з .NET;
- підтримка потужного засобу розробки – Visual Studio .NET;
- мовна незалежність у межах платформ, для яких реалізовано загальне мовне середовище виконання программ – CLR;
- можливості розширення за допомогою мультипроцесорних і кластерних рішень;
- нові можливості з обробки помилок;
- об’єктно-орієнтовані мови розробки (нова мова C#);
- розширені можливості повторного використання компонент.

Очевидно, що платформи .NET і ASP.NET надали нових можливостей для розробки Web-систем, задовольняють усім сучасним вимогам й дозволяють значно прискорити і спростити розробку складних прикладних програм. Однак на сьогодні .NET у повному обсязі наявне тільки для платформи Windows, а його перенесення на інші системи перебуває на стадії розробки, але ще не завершено, і майбутні результати оцінити важко. Оскільки ASP.NET сильно прив’язана до сервера IIS, хоча архітектура .NET дозволяє перенести прикладну програму ASP.NET на іншу платформу, нині реальної можливості розробляти сайти як кросплатформні рішення немає. Таким чином, найважливіша вимога до Web-орієнтованого програмного забезпечення – багатоплатформність поки що не може бути виконана платформою .NET, тобто її використання для створення Web-системи не завжди виправдано. Однак слід відзначити, що Web-система має інтегруватися з платформою .NET (передусім з Web-сервісами), оскільки її широке використання у майбутньому не викликає сумнівів.

Порівняння технологій. Розгляд найбільш популярних базових технологій побудови прикладних програм серверної сторони дозволяє назвати такі основні особливості їх архітектури:

– окреме виконання запитів, тобто під час кожного запиту динамічного вмісту запускається певна програма для його обробки, яка генерує вміст, що передається клієнтові,- цей підхід використовується у класичних CGI-скриптах;

– накопичення виконуваних процесів – підхід, аналогічний попередньому, але, якщо запит виконується повторно, нового запуску програми не відбувається, а обробка передається існуючому процесу,- такий підхід застосовують в технологіях Java Servlets, Fast CGI;

– шаблони сторінок, тобто під час запиту шаблони заповнюються динамічним вмістом, але необов'язково, вміст інтерпретується мовою сценаріїв, цей підхід застосовують в технологіях ASP, JSP, PHP;

– розширення Web-сервера,- який звертається до особливих розширень для опрацювання динамічного вмісту, причому ці розширення специфічні для Web-сервера, цей підхід використовують в ISAPI, NSAPI, mod_perl.

Кожен із зазначених підходів має свої переваги та недоліки і, відповідно, свою сферу застосування. Модель окремого виконання запитів істотно обмежує продуктивність. Варіант моделі з накопиченням процесів є розвитком технології побудови Web-орієнтованих прикладних програм, підвищує продуктивність, зберігаючи при цьому максимальну гнучкість розробки. Підхід, який використовує шаблони, надзвичайно зручний у разі розробки невеликих систем, однак зі збільшенням складності він починає гальмувати, тому процес розробки не підходить для великих систем. Він також вирізняється невисокою продуктивністю, хоча дослідження показують, що за певних умов може демонструвати досить високі показники і конкурувати з іншими підходами. Розширення Web-сервера не є найзручнішим засобом розробки, оскільки необхідно буде жорстко прив'язувати систему до певного Web-сервера, але система при цьому демонструє максимальну продуктивність і дає найбільшу гнучкість у процесі розробки прикладних сервісів.

За схемою обробки запитів платформи поділяють таким чином (технологію CGI не розглядається через незручність у використанні, низьку ефективність і те, що розширення серверів занадто сильно прив'язані до конкретних програмних продуктів):

– PHP-шаблони, під час виконання на Web-сервері Apache використовують як розширення сервера інтерпретатор (в експериментальному режимі можна також застосовувати сервер IIS);

– Java Servlets – накопичення процесів для кожного сервлета;

– JSP-шаблони – у процесі обробки виконується їх предкомпіляція в Java Servlets, що дозволяє використовувати схему накопичення процесів;

– ASP.NET шаблони - використовується схема попередньої компіляції, а не інтерпретації коду, для цього необхідною є наявність розширення Web-сервера IIS, також можна застосовувати і низькорівневі обробники.

Порівняльну оцінку основних характеристик платформ наведено у табл. 2.2, у якій «–» – немає підтримки; «–/+» – недостатня підтримка; «+/–» – підтримка не в повному обсязі; «+» – повна підтримка. Для порівняльних характеристик, таких як мова реалізації або продуктивність, оцінки відповідають ступеню переваги технології.

Таблиця 2.2. Порівняльна оцінка основних характеристик платформ

	PHP	**Java Servlets**	**JSP**	**ASP .NET**
Багатоплатформність	+/–	+	+	–/+
Продуктивність	–/+	+/–	+/–	+
Масштабованість	–	+	+	+
Мова реалізації	+/–	+	+	+
Можливості розширення й інтеграції	–	+	+/–	+
Простота використання, наявність засобів розробки	+/–	+/–	+	+
Наявність необхідних програмних бібліотек	+	+	+	+
Поділ дизайну і логіки	+/–	–/+	+/–	+
Засоби візуальної розробки	–/+	+/–	+	+
Можливість побудови компонентної архітектури	–	+	+/–	+

2.8.3. Технологія Rich Internet Applications

Нині у сфері створення Internet-прикладних програм визначилася тенденція до переходу від стандартних технологій до платформ, які дозволяють запускати в середовищі Web-браузера програми, які за зовнішнім виглядом і поведінкою не відрізняються від настільних (desktop) програм. Такий вид програмних продуктів називають Rich Internet Applications (RIA). Незважаючи на те, що деякі з технологій RIA використовують досить давно (з кінця 90-х років), найбільшого поширення вони набули останнім часом.

Зазвичай RIA прикладна програма:

– передає Web-клієнтові необхідну частину призначеного для користувача інтерфейсу, залишаючи велику частину даних (ресурси програми, дані, тощо) на сервері;

– запускається у браузері й не вимагає установки додаткового програмного забезпечення;

– запускається локально в середовищі обчислювального процесу, в якому гарантовано безпеку, названого «пісочниця» (sandbox).

За допомогою діаграми Венна на рис. 2.52 продемонстровано можливі області застосування RIA як технології побудови програмних систем.

Рис. 2.52. Можливі області застосування RIA як технології побудови програмних систем

Традиційні Web-прикладні програми переважно працюють на сервері – у клієнтській програмі-браузері (Internet Explorer, Firefox або Opera) відображається контент, написаний мовою розмітки HTML. Згідно з цим підходом традиційні Web-прикладні програми мають такі недоліки у процесі функціонування:

мають обмежений рівень інтерактивності; постійно взаємодіють із сервером – надсилають на нього дані, отримують відповіді й завантажують нові сторінки.

У RIA значну частину функціонала винесено у клієнтську частину, тобто низка функцій виконується на стороні клієнта (у тому ж браузері, але з використанням плагинів і/або технології Ajax), що дозволяє отримати прикладну програму, яка має такі переваги: працює швидше за рахунок меншої кількості звертань до сервера і того, що немає перевантажень сторінок; забезпечує близький до традиційних настільних прикладних програм рівень інтерактивності та зручності інтерфейсу (наприклад, drag&drop).

Розглянемо архітектурні особливості побудови RIA–прикладних програм. Порівняння принципів роботи Web-прикладної програми і RIA-прикладної програми зображено на рис. 2.53.

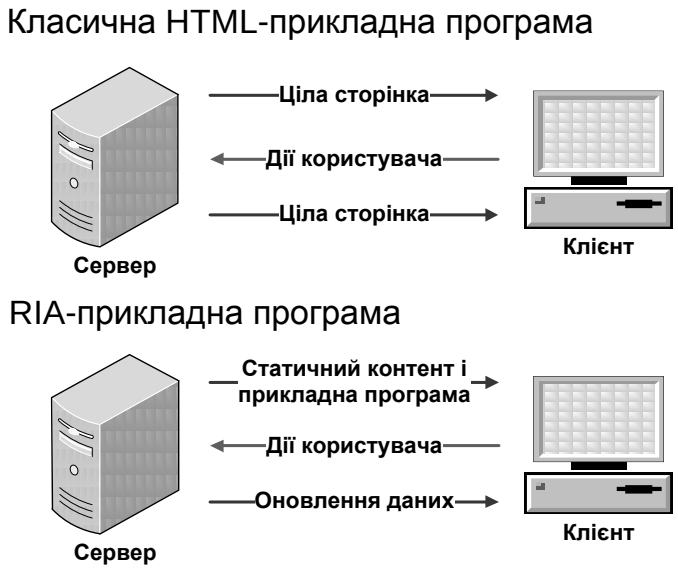

Рис. 2.53. Принципи роботи традиційної Web-прикладної програми і RIA-прикладної програми

Порівняльні дані профілізації традиційної Web-прикладної програми (на прикладі сайту mail.ru) і RIA-прикладної програми (на прикладі сайту gmail.com) подано у табл. 2.3.

Таблиця 2.3. Порівняння даних профілізації Web- і RIA-прикладних програм

Дія	**mail.ru**	**gmail.com**
Початкова сторінка	24 запити 116 КБайт	33 запити 379 КБайт
Відкрити лист	17 запитів 71 КБайт	3 запити 2 КБайт

Також можна порівняти архітектуру товстого клієнта і RIA-прикладної програми.

Товстий клієнт – це високопродуктивна інтерактивна прикладна програма з багатим, призначеним для користувача, інтерфейсом, яка працює в різних сценаріях незалежно від наявності Internet.

Зазвичай архітектуру розглядають за такими компонентами (рис. 2.54):

– рівень подання – відповідає за взаємодію прикладної програми з користувачем, формує і перевіряє введені користувачем дані;

– рівень бізнес-логіки – охоплює бізнес-потоки й логіку, які власне і керують прикладною програмою; також на цьому рівні перебувають бізнес-сутності, тобто елементи, якими обмінюються компоненти логіки;

– рівень даних – містить моделі та провайдери, які дозволяють отримувати дані.

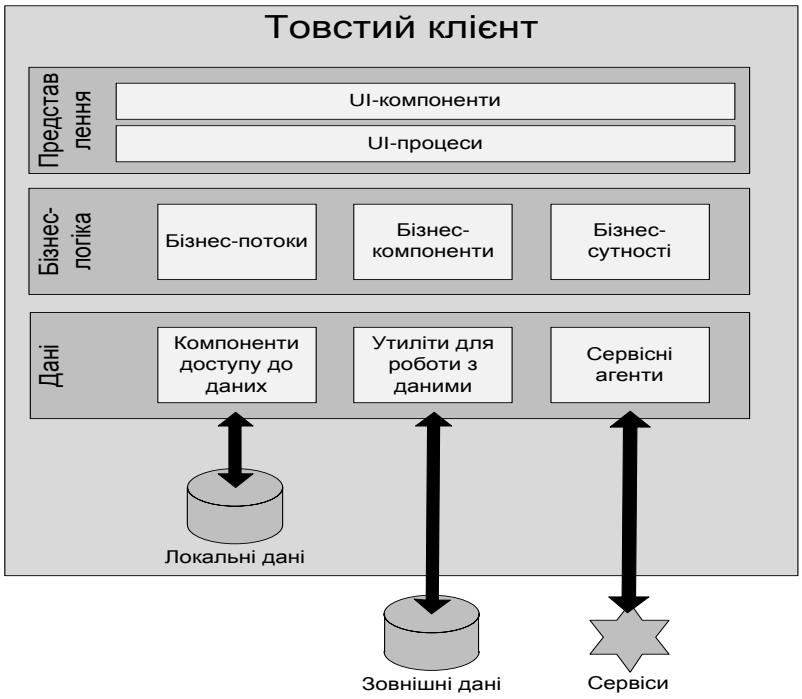

Рис. 2.54. Архітектура товстого клієнта

Internet-прикладні програми RIA виконуються в «пісочниці» браузера. Архітектура Web-сервера має такі компоненти (рис. 2.55):

– рівень сервісів надає інтерфейси для клієнтської частини для взаємодії з рівнем логіки;

– рівень бізнес-логіки ідентичний рівню логіки товстого клієнта;

– рівень даних ідентичний рівню даних товстого клієнта.

Порівняння архітектури програмного забезпечення за технологією товстого клієнта й архітектури RIA-прикладної програми подано у табл. 2.4.

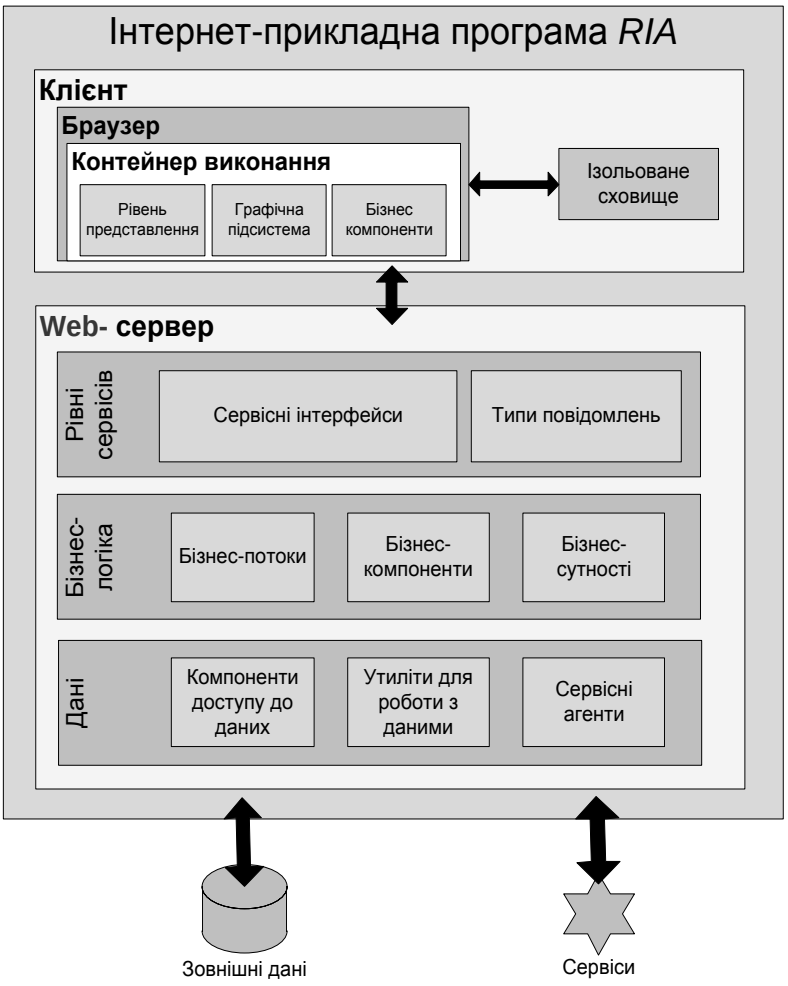

Рис. 2.55. Архітектура RIA-прикладної програми

Таблиця 2.4. Порівняння архітектури програмного забезпечення за технологією товстого клієнта й архітектури RIA-прикладної програми

Архітектура	**Переваги**	**Проблеми**
Товстий клієнт	1. Може витрачати клієнтські ресурси; 2. інтерактивність, багатий призначений для користувача функціонал; 3. висока здатність до реагування; 4. підтримка роботи без мережі.	1. Вимагає установки (Click Once, Windows Installer, Copy); 2. контроль версій неефективний або відсутній; 3. залежність від платформи
RIA	1. Такий самий багатий, призначений для користувача, інтерфейс, як і у технології товстого клієнта; 2. підтримка 3D і медіа високої якості; 3. простий спосіб поширення; 4. простий механізм контролю версій; 5. крос-платформність і крос-браузерність.	1. Невеликий розмір прикладних програм; 2. обмеження щодо використання клієнтських ресурсів; 3. вимагає встановленого плагіна для браузера.

Узагальнюючи, зауважимо, що RIA-прикладні програми залучили:

від товстого клієнта:

- надання користувачеві інтерактивного інтерфейсу;
- швидкий час відгуку інтерфейсу без оновлення сторінки;
- звичні засоби призначеного для користувача інтерфейсу, зокрема drag&drop і можливість роботи в режимі онлайн та оффлайн;

від Internet-прикладної програми:

- швидке розгортання;
- крос-платформність;
- використання прогресивного завантаження для отримання інформації і даних;
- використання більшості Internet-стандартів;

від засобів передачі даних:

- інтерактивні аудіо- і відеотехнології.

Розглянемо технології створення RIA-прикладних програм, які використовують на сьогодні.

Технології Adobe. Фірма Adobe послідовно просуває свої RIA-технології на основі Flash-платформи і надає повний стек рішень та інструментів для створення візуальної та медійної складових, а також серверних рішень. Flash-плеєр доступний на всіх основних платформах і в усіх основних браузерах. Окрім того, наявна версія для мобільних пристроїв. Стек технологій Adobe для створення RIA-прикладних програм виглядає, як показано на рис. 2.56.

Adobe Flash. Adobe Flash (раніше відома як Macromedia Flash) – мультимедійна платформа, використовувана для створення векторної анімації та інтерактивних прикладних програм (у тому числі, ігор), а також для інтеграції відеороликів у Web-сторінки.

Adobe Flash дозволяє працювати з векторною, растровою та обмежено з тривимірною графікою, а також підтримує двонаправлену потокову трансляцію аудіо і відео. Для кишенькових персональних комп'ютерів (КПК) й інших мобільних пристроїв випущена спеціальна полегшена версія платформи Flash Lite, функціональність якої обмежена з погляду можливостей мобільних операційних систем і їх апаратних характеристик.

Adobe Flex. Adobe Flex також дозволяє розробляти прикладні програми на базі Flash-платформи, але без використання часової шкали, замість якої пропонується мова розмітки MXML, що дає можливість у декларативній формі задавати зміни станів і переходи між ними. Adobe Flex призначено виключно для розробників, оскільки з його допомогою створюють складні Internet-прикладні програми для бізнес-завдань і для щоденного застосування. Уся розробка у Flex орієнтована на використання готового набору розширюваних компонентів, що підходять майже для всіх завдань. Таблиці стилів CSS дозволяє гнучко настроювати зовнішній вигляд як окремих компонентів, так і всієї прикладної програми. Зв'язування даних допомагає відображувати змінену інформацію фактично без єдиного рядка коду.

Flex-прикладна програма може компілюватися на сервері (для цього буде потрібно *mod_flex.so* або *mod_flex.dll* залежно від ОС і Web-сервера) або з IDE, або безпосередньо з командного рядка за допомогою компілятора

mxmlc (починаючи з Flex 2). Як і у Flash, результатом є файл swf, виконуваний у Flash Player. Принцип роботи Flex показано на рис. 2.57.

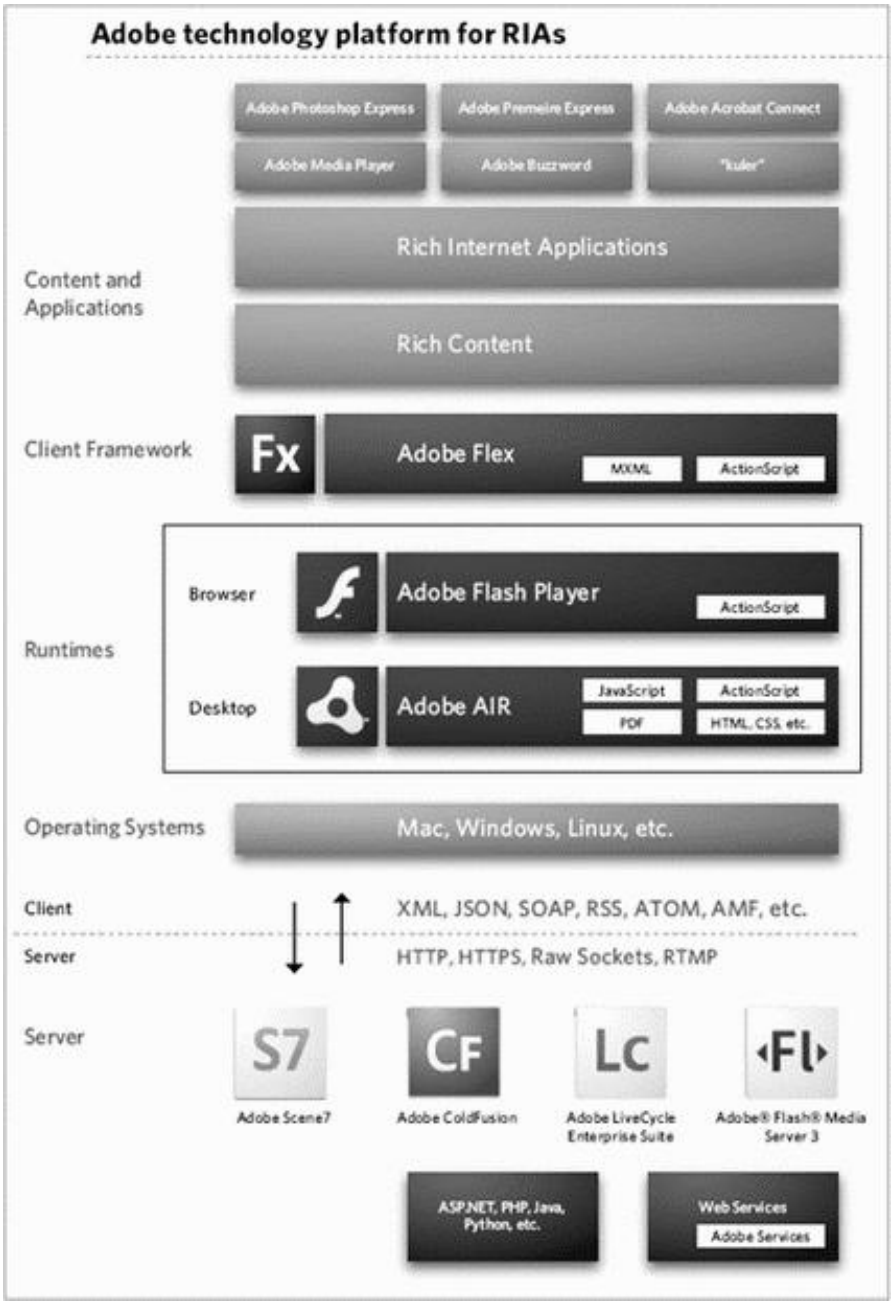

Рис. 2.56. Стек технологій Adobe для створення RIA-прикладних програм

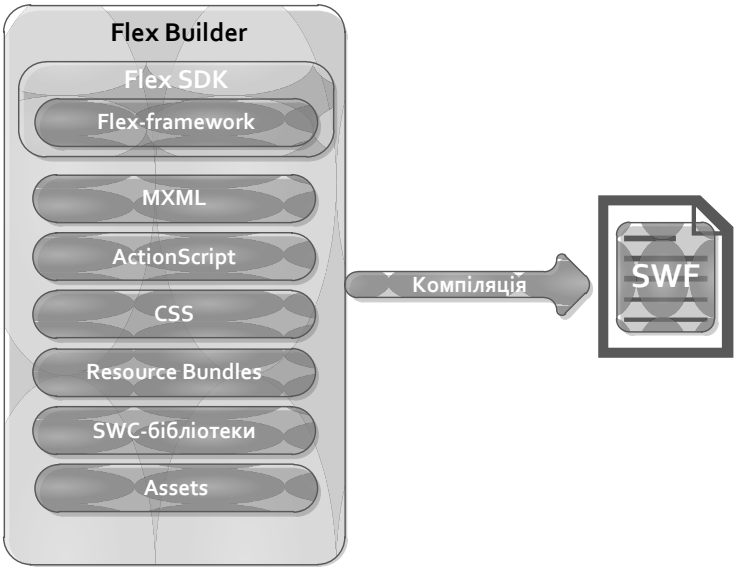

Рис. 2.57. Принцип роботи Flex

Flex – це великий набір класів (Flex SDK), що містить багато компонентів, які не увійшли у Flash. Позитивні якості Flex зумовлені його специфікою, зокрема мовою MXML, наприклад, Flex-компоненти для побудови графіків є зручними і виглядають дуже прийнятно для використання у стилі «як є».

Adobe Air. Adobe Air (Adobe Integrated Runtime) дозволяє запускати Web-прикладні програми на основі як HTML, так і Flash-платформи з можливістю перегляду PDF-документів на «робочому столі» персонального комп'ютера. Таким чином, розробники Web-орієнтованого програмного забезпечення можуть створювати настільні прикладні програми, використовуючи вже наявні типові шаблони прикладних компонентів. Adobe AIR надає низку можливостей щодо інтеграції з операційною системою: вікна, доступ до файлової системи, захищене локальне сховище, локальну базу даних і технологію перетягання об'єктів (drag&drop). AIR-прикладні програми можуть встановлюватися прямо з Internet. Архітектуру платформи AIR подано на рис. 2.58.

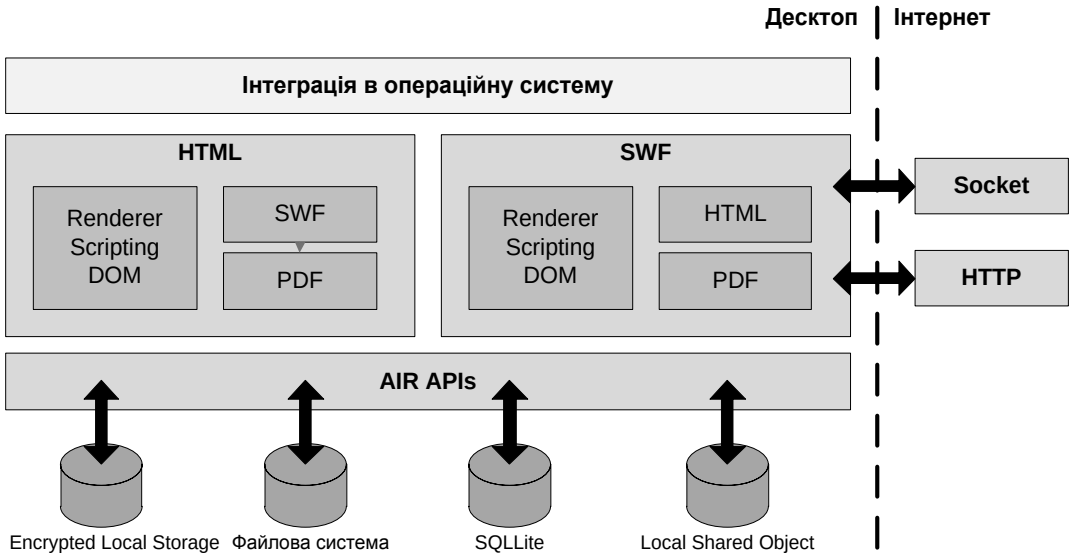

Рис. 2.58. Архітектура платформи AIR

Технологія Ajax. Ajax є поєднанням технологій, в основі яких лежать мова JavaScript і об'єкт XMLHttpRequest, завдяки чому прикладні програми можуть отримувати дані з сервера і відображати їх без перезавантаження сторінки. Формально Ajax не є RIA-технологією, оскільки не інтегрує в собі медіа можливостей, не має стандартних засобів промальовування векторної графіки, а JavaScript по-різному виконується в різних браузерах.

Ці проблеми намагаються вирішити за допомогою Ajax-бібліотек, але вони ґрунтуються на гібридних технологіях (наприклад, використання Flash для відео). Для подолання проблеми крос-браузерності також знайдено рішення, але вона знову може виникнути у разі виходу нових версій браузерів.

Нині є безліч Ajax-бібліотек, що мають свої особливості й сфери застосування, проте немає єдиного стандарту, тому, переходячи з однієї бібліотеки на другу розробник вимушений повторно вивчати документацію.

Використання Ajax дозволяє суттєво збагатити традиційні Web-сторінки і зробити користування ними дуже комфортним. Застосовувати Ajax, створюючи сайти-презентації та Internet-прикладні програми, які максимально наближені до рівня настільних прикладних програм, є недоцільним.

Основні переваги технології Ajax такі:

- наявна в будь-якому браузері;
- проста в освоєнні;

- швидкий цикл розробки;
- розвинений інструментарій;
- розвинене співтовариство.

Основні недоліки технології Ajax такі:

- іноді наявна крос-браузерна несумісність;
- складно/дорого розробляти великі прикладні програми;
- складно налагоджувати, дорого підтримувати;
- низька продуктивність.

Технології Microsoft. Корпорація Microsoft враховуючи актуальність RIA-технологій пропонує розробникам низку засобів створення насичених Internet-прикладних програм.

ActiveX. Технологія ActiveX – засіб, за допомогою якого Internet Explorer (IE) використовує інші прикладні програми всередині себе, завантажує Windows Media Player, Flash, Quicktime та інше прикладне програмне забезпечення, яке може відтворювати файли, вбудовані у web-сторінки. Елементи керування ActiveX активізуються, якщо клацнути по такому об'єкту на web-сторінці, наприклад, WMV-файлу, щоб завантажити його для відображення у вікні браузера IE.

Як продукт Microsoft ActiveX була спеціально спроектована для роботи із системами Windows, тому не підтримується іншими операційними системами, такими як Mac або Linux.

Firefox та інші крос-платформні браузери використовують систему «Програмний інтерфейс модулів Netscape» (Netscape Plugin Application Programming Interface, NPAPI), яка під'єднується та виконує функції, подібні до ActiveX.

Керівні елементи ActiveX – еквівалент модулів NPAPI, проте ActiveX і технологія модулів типу NPAPI відмінні. Наприклад, за допомогою надбудови ActiveX можна завдати шкоди комп'ютеру, модулі NPAPI мають кращі функції безпеки, менше контролюють web-сторінку, в яку вони вбудовані, та не можуть використовуватися у прикладних програмах, відмінних від web-браузеру. ActiveX має ширше застосування для великої кількості прикладних програм.

Як технологія для RIA технологія ActiveX має низку суттєвих недоліків:

- платформа тільки IE і Windows;
- нетривіальна у розробці та відладці;
- великий період освоєння технології;
- є деякі проблеми з безпекою.

Основною перевагою технології ActiveX для RIA є те, що вона дозволяє вбудовувати компоненти в різні системи без додаткового програмування.

Технологія WPF. Windows Presentation Foundation (WPF, кодова назва Avalon) – графічна (презентаційна) підсистема у складі .NET Framework 3.0, що використовує мову програмування XAML (Extensible Application Markup Language), яка дозволяє створювати широкий спектр програмних мультимедіа інтерфейсів. Технологія WPF разом з .NET Framework 3.0 працює у Windows Vista і Windows 7, а також їх можна установити у Windows XP SP2 і Windows Server 2003 і 2008.

Технологічне середовище у складі WPF і .NET Framework 3.0 призначено для користувача інтерфейсу, містить нове ядро, яке має замінити GDI і GDI+, використовувані в нинішній Windows-платформі. Технологія WPF є високорівневим об'єктно-орієнтованим функціональним шаром (framework), що дозволяє створювати 2D- і 3D-інтерфейси, у майбутньому має об'єднати Windows і Web-розробку (в тому числі, AJAX), для подання на інтерфейсі користувача використовує не GDI+, а DirectX, підтримує теми і нестандартні для WinForms контроли, її продуктивність вища, ніж у GDI+ за рахунок використання відеокарти.

Технологія Silverlight. Microsoft Silverlight надає графічну систему, схожу на Windows Presentation Foundation, і об'єднує мультимедіа, графіку, анімацію та інтерактивність в одній програмній платформі. Технологію було розроблено, щоб працювати з мовою розмітки сторінок XAML, що використовує векторну графіку й анімацію, а також із мовами Microsoft .NET. Текст, що міститься в Silverlight-прикладних програмах, доступний для пошукових систем у вигляді XAML, оскільки він не компілюється. Технологію Silverlight також можна використовувати для того, щоб створювати widget для Windows Sidebar у операційній системі Windows.

Технологія Silverlight дозволяє створювати плагіни для браузера, що запускають прикладні програми, які містять анімацію, векторну графіку та аудіо-, відеоролики, що характерно для RIA. Версія Silverlight 2.0 додала підтримку для мов .NET й інтеграцію з IDE, наступні версії (Silverlight 3, майбутня Silverlight 4) матимуть більш потужні функціональні можливості.

Silverlight підтримує відтворення WMV, WMA і MP3 для всіх підтримуваних браузерів, не вимагаючи додаткових компонентів, зокрема Windows Media Player. Silverlight дозволяє динамічно завантажувати XML-файли і використовувати програмний інтерфейс для доступу до документів DOM для взаємодії з ними аналогічно до Ajax; містить об'єкт Downloader, завдяки якому можна завантажувати скрипти, медіафайли тощо, якщо це необхідно прикладній програмі. Починаючи з версії 2.0, логіка програми може бути описана будь-якою з мов .NET, включаючи динамічні мови програмування Iron Ruby та Iron Python, які, у свою чергу, виконуються в *DLR* (*Dynamic Library Runtime*), а не *CLR*.

Технології Oracle (Sun). Розглянемо також *RIA*-технології компанії Sun Microsystems, яка входить до складу Oracle Corporation.

Технологія *JavaFX*. *JavaFX* – це платформа для створення *RIA*, які можуть запускатися на персональних комп'ютерах і мобільних пристроях. Така технологія дозволяє створювати прикладні програми для роботи з мультимедійним контентом, графічні інтерфейси користувача для бізнес-прикладних програм, ігри для персональних комп'ютерів і мобільних пристроїв, насичені графікою, мультимедіа web-сайти тощо.

Прикладна програма *JavaFX* створюється за допомогою декларативної мови програмування *JavaFX Script*, для розробки якої необхідно використовувати *JavaFX 1.0 SDK* (*Software Development Kit*).

З коду, написаного мовою *JavaFX Script*, можна звертатися до будь-яких бібліотек *Java*, тому спільне використання мов *Java* і *JavaFX Script* дозволяє вирішувати різноманітні завдання, наприклад, бізнес-логіка прикладних програм може бути написана на *Java*, а графічний інтерфейс користувача на *JavaFX Script*.

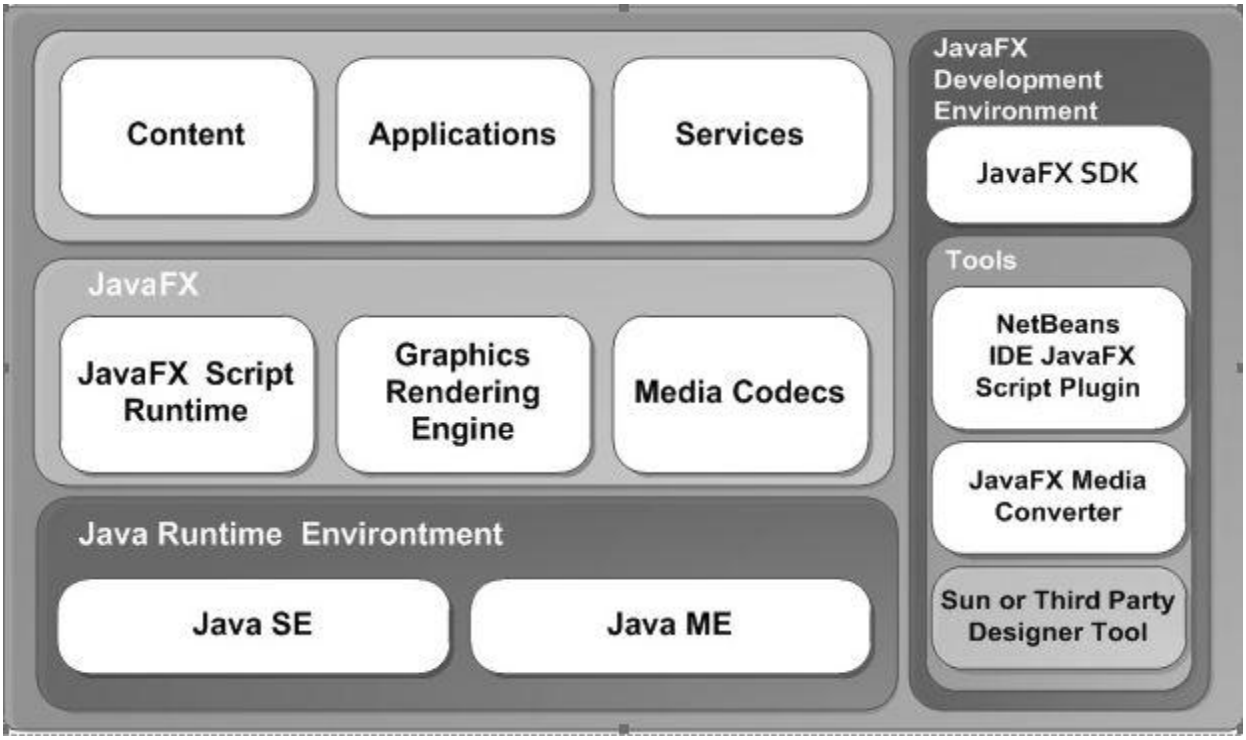

Рис. 2.59. Компоненти для створення RIA-прикладних програм

Прикладні програми, написані мовою Java FX Script, можуть запускатися на комп'ютерах зі встановленим середовищем виконання Java 1.5 і вище, нині їх виконання підтримують наступні операційні системи: Windows, Mac OS x, GNU/Linux і Solaris.

Інші технології. Наявні інші технології створення RIA-прикладного програмного забезпечення, однією з яких є Google Gears (модуль розширення для браузеру), який дозволяє AJAX-прикладним програмам частково усунути «пісочниці». Основні можливості Google Gears такі:

- доступність ресурсів offline;
- локальна SQLite база даних, синхронізація даних із сервером;
- асинхронні завдання;
- вибір декількох файлів для завантаження;
- попередній перегляд у реальному часі картинок;
- наявність технології drag&drop.

Проте восени 2009 року компанія Google зупинила розвиток технології Google Gears на користь HTML 5.0, продовжуючи тільки її підтримку.

Специфікація HTML 5.0, що перебуває на стадії розробки, підтримує такі елементи RIA:

- малювання 2D-картинок у реальному часі;
- контроль над програванням медіафайлів;
- зберігання даних у браузері;
- використання технології drag&drop;
- роботу з мережею, push-технологію (технологію просування контенту).

Порівняння вищеописаних технологій. Порівняння технологій створення RIA-прикладного програмного забезпечення наведено у табл. 2.5.

Таблиця 2.5. Порівняння технологій створення RIA-прикладних програм

Технології	**WPF**	**Silverlight**	**Flex**	**Flash**	**Adobe Air**	**JavaFX**
Основні характеристики						
ОС	Windows	Windows MacOS Linux	Windows MacOS Linux	Windows MacOS Linux	Windows MacOS Linux	Windows MacOS Linux
Браузери	IE, Firefox	IE, Firefox, Safari, Chrome	IE, Firefox, Safari, Opera, Netscape, Chrome	IE, Firefox, Safari, Opera, Netscape, Chrome	–	IE, Firefox, Safari, Opera, Netscape, Chrome
Відсутність потреби в ліцензуванні	+/–	–	+/–	–	+/–	+
Розробка						
Мови	XAML+ CLS (C#, VB)	XAML+ JavaScript/ CLS(C#, VB)/DLR (Python)	MXML+ ActionScript	ActionScript	MXML/ ActionScript /JavaScript/ HTML	JavaFX Script, Java platform (Java, Groovy)

Продовження табл. 2.5

Технології	**WPF**	**Silverlight**	**Flex**	**Flash**	**Adobe Air**	**JavaFX**
Data Binding	+	+	+	–	+	+
Декларативна мова розмітки	+	+	+	–	+	+
Змішування розмітки і коду	+	+	+	–	+	+
Code Behind	+	+	+	–	+	+
Інструментарій						
Інструментарій	Visual Studio, Microsoft Expression (Blend), XamlPad, Notepad.	Visual Studio, Microsoft Expression (Blend), Notepad	Flex Builder, Thermo, Notepad.	Flash, Flash Develop, FDT	Flex Builder, Aptana Studio, Notepad	NetBeans, Eclipse, JavaFX Pad, JFXBuilder, Notepad
WYSIWYG - редактори	Expression Blend, Visual Designer for WPF	Expression Blend	Flex Builder, Thermo	Flash	Flex Builder, Thermo, Flash	JFXBuilder
Платформи для розробки	Windows	Windows	Windows MacOS Linux	Windows MacOS	Windows MacOS Linux	Windows MacOS Linux
Вартість SDK	Безкоштовно	Безкоштовно	Безкоштовно	платна	Безкоштовно	Безкоштовно

Продовження табл. 2.5

Технології	**WPF**	**Silverlight**	**Flex**	**Flash**	**Adobe Air**	**JavaFX**
Фреймворки (каркаси)						
GUI - фреймворк	Компоненти і layout	Компоненти layout	Компоненти і layout	Компоненти	Компоненти і layout	Компоненти і layout
Розширюваність	+	+	+	+	+	+
Технологія drag&drop	+	+	+	–	+	+
Компоненти сторонніх розробників	+	+	+	+	+	–
Підтримка accessibility	+	+	+	+	+	+
GUI Automation	+	+	+	+	–	+
Кастомізація/скінизация (використання алгоритмів кешування)						
Кастомізація зовнішнього вигляду компонент	+	+	+	+	+	+
Designtime skinning (styling)	+	+	+	+	+	+
Runtime skinning (styling)	+	+	+	–	+	+
Відео						
Кодеки	(Windows Media Player)	WMV, VC - 1, H.264, ACC Video, DRM	Sorenson Spark, On2 VP6 - E, On2 VP6 - S, H.264, flv	Sorenson Spark, On2 VP6 - E, On2 VP6 - S, H.264, flv	Sorenson Spark, On2 VP6 - E, On2 VP6 - S, H.264, flv	–

Продовження табл. 2.5

Технології	**WPF**	**Silverlight**	**Flex**	**Flash**	**Adobe Air**	**JavaFX**
Hardware Acceleration	+	+	+	+	+	-
HD-відео	+	+	+	+	+	-
DRM	+	+	–	–	–	-
Streaming	+	+	+	+	+	-
Звук						
Формати	Windows Media Player	MP3, WMA, DRM	MP3, AAC	MP3, AAC	MP3, AAC	AU, AIFF, WAV, MIDI
VoIP	–	–	–	–	–	–
Наявність VoIP-рішень	+	–	+	+	+	–
Графіка						
Вбудована підтримка 3D	+	+	–	–	–	–
Векторна графіка	+	+	+	+	+	+
Завантаження JPG	+	+	+	+	+	+
Завантаження PNG	+	+	+	+	+	–
Завантаження GIF	+	–	+	+	–	+
Hardware acceleration	+	+	–	–	–	+
Бітмап-ефекти	+	+	+	+	+	+

Закінчення табл. 2.5

Технології	WPF	Silverlight	Flex	Flash	Adobe Air	JavaFX
Додатково						
Підтримка мікрофона	+	–*	+	+	+	–
Підтримка вебкамери	+	–*	+	+	+	–
Повноекранний режим	+	+	+	+	+	–
Клієнт/ сервер протоколи	.NET-сумісне програмне забезпечення	XML (JSON, Web services, RSS, POX, REST), WCF Services	HTTP, Socket, XML, AMF, RTMP	HTTP, Socket, XML, AMF, RTMP	HTTP, Socket, XML, AMF, RTMP	Програмне забезпечення, яке виконується на Java-платформі

* Очікується у версії Silverlight 4.0.

Колонка для WPF стосується XBAP, а колонка JavaFX-аплетів на основі JavaFX, оскільки розглянуті види прикладних програм орієнтовані на Web, а JavaFX- і WPF-прикладні програми можуть бути, а можуть і не бути пов'язаними з Web.

2.8.4. Дескриптор розгортання web-прикладних програм та компонент

Дескриптор (англ. Descriptor) – дослівно описувач, описовий елемент. **Дескриптор** – невід'ємне ціле число, яке задає номер будь-якого ресурсу в процесі роботи з ним і використовується зазвичай через деякий інтерфейс, причому смисл значення дескриптора схований за цим інтерфейсом.

Платформа Java 2 Enterprise Edition. J2EE-платформа – це розподілена комп'ютерна платформа, що полегшує дизайн, розробку, компонування і розгортання компонентно-орієнтованих корпоративних прикладних програм.

Розрізняють безліч способів, за якими можна подавати специфічний набір бізнес-вимог, логічну модель і опис рішення, які задовольняють таким вимогам. J2EE-платформа найбільш придатна і найкраще підходить для підтримування крос-платформності, безпеки, транзакційних прикладних програм, які надають корпоративну інформацію для Internet- та Intranet-клієнтів.

J2EE-прикладна програма являє собою набір програмних компонентів, створених для поширення на кілька рівнів. J2EE-платформа сервер-орієнтована, тому для J2EE-прикладних програм типово пропонувати засоби різноманітним клієнтам. Терміни «клієнт» і «сервер» використовують, щоб відобразити логічну, а не апаратну структуру.

Не потрібно, щоб J2EE-прикладна програма була розділеною, але вона має бути простою у використанні різними типами комп'ютерних систем, якщо це потребують ділові або технічні вимоги. Процеси розробки, компонування та поширення логічно розділені всередині середовища J2EE. Процес компонування програми керується і проводиться за рахунок дескрипторів, що містять декларативні визначення, які в підсумку формують поведінку прикладних програм у специфічному операційному середовищі.

Платформа J2EE призначена передусім для розробки розподіленого Web-орієнтованого прикладного програмного забезпечення і підтримує такі види компонентів:

Enterprise JavaBeans (EJB). Компоненти EJB призначені для реалізації на їх основі бізнес-логіки програми та операцій над даними. Будь-які компоненти, розроблені на Java, називають бінами.

Web-компоненти (Web components). Ці компоненти надають інтерфейс до корпоративних програмних систем зверху широко використовуваних протоколів Internet, зокрема HTTP. Надані інтерфейси можуть бути як інтерфейсами для людей (WebUI), так і спеціалізованими програмними інтерфейсами, що працюють подібно віддаленому виклику методів, але зверху HTTP. До групи Web-компонентів входять фільтри (filters), обробники Web-подій (web eventlisteners), сервлети (servlets) і серверні сторінки Java (JSP).

Звичайні програми на Java. J2EE (Java Platform Enterprise Edition) є розширенням J2SE (Java Platform Standart Edition), тому всі прикладні програ-

ми, написані мовою Java, можуть працювати і в цьому середовищі. Однак, окрім звичайних можливостей J2SE, ці прикладні програми можуть використовувати у своїй роботі Web-компоненти і EJB як безпосередньо, так і віддалено, зв'язуючись з ними по HTTP.

Аплети (applets). Це невеликі компоненти, що мають графічний інтерфейс користувача і призначені для роботи всередині стандартного Web-браузера. Вони використовуються тоді, коли не вистачає можливостей подання інформації для користувача інтерфейсу на основі HTML, і можуть зв'язуватися з віддаленими Web-компонентами, які працюють на сервері, по HTTP.

Компоненти, які підтримує ця платформа, мають дескриптор розгортання (deployment descriptor) – опис у встановленому форматі на основі XML- конфігурації компонента у межах контейнера, в якому він міститься. Прикладна програма в цілому також має дескриптор розгортання. Дескриптори розгортання відіграють важливу роль, дозволяючи змінювати деякі параметри функціонування компонента і прив'язувати їх до параметрів середовища, у межах якого компонент працює, не змінюючи його коду.

Розглянемо способи вирішення спільних завдань побудови розподілених систем на основі платформи J2EE.

Цілісність і несуперечність даних під час роботи J2EE-прикладних програм підтримується за допомогою механізму розподілених транзакцій, керувати якими може EJB-контейнер, який створюється визначенням політики участі методів EJB-компонентів у транзакції в їх дескрипторах розгортання або може здійснюватися вручну. В обох випадках використовуються механізми, що реалізують інтерфейси керування транзакціями Java (Java Transaction API, JTA).

Базові інтерфейси JTA містяться у пакетах *javax.transaction* і *javax.transaction.xa*. Це, передусім, інтерфейси менеджера транзакцій *TransactionManager*, самих транзакцій *Transaction* і *UserTransaction* та інтерфейс синхронізації *Synchronization*, що дозволяє отримувати повідомлення про початок завершення і кінець завершення транзакцій.

Методи інтерфейсу *TransactionManager* дозволяють запустити транзакцію, завершити її успішно або відкотити, а також отримати об'єкт, що подає поточну транзакцію і має тип *Transaction*. Методи інтерфейсу *Transaction* дають

змогу завершити або відкотити транзакцію, яку подає об'єкт такого інтерфейсу, зареєструвати об'єкти для синхронізації у процесі завершення транзакції, а також додати деякі ресурси до числа учасників даної транзакції або видалити їх із цього списку. Такі ресурси подаються у вигляді об'єктів інтерфейсу *javax.transaction.xa.XAResource*.

Інтерфейс *UserTransaction* використовують, щоб керувати призначеними для користувача транзакціями – він надає дещо менше можливостей, ніж *TransactionManager*.

У тому разі, якщо керування транзакціями цілком доручається EJB-контейнеру (це транзакції, керовані контейнером, *container managed transactions*), впливати на їх перебіг можна, зазначаючи в дескрипторах розгортання EJB-компонентів різні транзакційні атрибути (*transaction attributes*) для їх методів. Транзакційний атрибут може набувати одною з таких значень:

Required. Метод, у якого наявний такий атрибут, завжди має виконуватися в контексті транзакції, тобто він працюватиме в контексті тієї самої транзакції, в якій працював метод, що викликав його, а якщо він був викликаний поза контекстом транзакції, з початком його роботи буде запущена нова транзакція. Цей атрибут використовується найбільш часто;

RequiresNew. Метод, який має такий атрибут, завжди буде запускати нову транзакцію на самому початку роботи, при цьому зовнішня транзакція, якщо вона виконувалась, буде тимчасово припинена;

Mandatory. Метод, у якого наявний такий атрибут, має викликатися тільки з транзакції, у контексті якої він і продовжить працювати. У разі виклику такого методу ззовні транзакції буде створена виняткова ситуація, зокрема *TransactionRequiredException*;

NotSupported. У разі виклику такого методу зовнішня транзакція, якщо вона є, буде тимчасово припинена, а якщо її немає, то нова транзакція не буде запущена;

Supports. Такий метод працює у транзакції, якщо його викликали з її контексту, якщо ж він був викликаний поза транзакцією, то нова транзакція не запускається;

Never. У разі виклику такого методу з транзакції створюється виняткова ситуація *RemoteException*. Він може працювати, тільки якщо його викликано ззовні транзакції.

Відкотити автоматично керовану транзакцію можна, створивши виняткову ситуацію *javax.ejb.EJBException* або викликавши метод *setRollbackOnly()* інтерфейсу *javax.ejb.EJBContext*.

Захищеність J2EE-програми підтримується декількома способами:

1. За допомогою методів аутентифікації, тобто визначення ідентичності користувачів, які позначають у дескрипторі розгортання програми. Можна використовувати такі способи аутентифікації:

– аутентифікація не виконується;

– за допомогою основного механізму протоколу HTTP. У разі спроби звертання до ресурсу за протоколом HTTP буде запитане ім'я користувача та пароль, які перевірить Web-сервер. Цей спосіб не дуже добре захищений, оскільки реквізити користувача пересилаються мережею в незашифрованому вигляді;

– за допомогою дайджесту (digest). Цей метод працює так само, як основний механізм аутентифікації по HTTP, але ім'я і пароль користувача пересилаються в зашифрованому вигляді. Такий спосіб використовується досить рідко;

– за допомогою спеціальної форми. У такому разі використовують сторінку, на якій розміщена форма аутентифікації (зазвичай це ті самі поля для введення імені користувача та пароля, але, можливо, і якихось інших його атрибутів), і сторінка, на якій міститься повідомлення, що видається у разі невдалої аутентифікації;

– з використанням сертифіката клієнта. Цей метод використовує протокол HTTPS, клієнт повинен надати свій сертифікат або відкритий ключ, який відповідає стандарту X.509 на інфраструктуру відкритих ключів. Можна використовувати і взаємну аутентифікацію – у цьому разі й клієнт, і сервер надають свої сертифікати.

2. За допомогою з'єднань за протоколом HTTP поверх рівня захищених сокетів (Secure Socket Layer, SSL, на HTTP поверх SSL часто посилаються за допомогою окремої абревіатури HTTPS). Можна використовувати лише такі комбінації, вказавши атрибути *CONFIDENTIAL* та/або *INTEGRAL* у де-

скрипторі розгортання програми. Перший атрибут означає, що передачу даних між клієнтом та програмою буде зашифровано так, що їх важко буде прочитати третій стороні. Другий атрибут означає, що ці дані супроводжуватимуть додатковою інформацією, яка гарантує їх цілісність, тобто те, що їх не було підмінено десь між сторонами, які беруть участь у зв'язку.

3. За допомогою механізму опису ролей визначення доступності різних методів Web-компонентів і EJB-компонентів для різних ролей, а також завдання політики перенесення або створення ролей під час спільної роботи кількох методів. Ролі, політики їх перенесення і правила доступу різних ролей до методів описуються в дескрипторах розгортання компонентів. У процесі розгортання програми зареєстровані на J2EE-сервері користувачі та групи користувачів можуть бути відображені на різні ролі, зазначені в прикладній програмі.

4. За допомогою встановлення обмежень доступу до наборів ресурсів, що задаються у вигляді списків уніфікованих ідентифікаторів ресурсів (URI) або шаблонів URI. Ці обмеження описуються в дескрипторі розгортання програми та визначають ролі й дозволені їм види прямого доступу (не через звертання до інших компонентів) до певного набору URI.

5. За допомогою програмного визначення ролей і користувачів, від імені яких працює поточний потік, з коду самих компонентів. Це можна робити за допомогою методів *isUserInRole()* та *getUserPrincipal()* інтерфейсу HTTP ServletRequest, використовуваного для подання запитів до Web-компонентів, й аналогічних методів *isCallerInRole()* та *getCallerPrincipal()* інтерфейсу EJBContext, для опису контексту виконання методів EJB-компонентів.

Платформа .NET. Середовище .NET призначено для більш широкого використання, ніж платформа J2EE. Однак його функціональність у частині, призначеній для розробки розподілених Web-прикладних програм, дуже схожа на J2EE. Роль дескрипторів розгортання відіграють конфігураційні файли, подані в певному форматі на основі XML.

EJB-контейнер. У цілому EJB-контейнер являє собою приклад об'єктного монітора транзакцій – програмного забезпечення проміжного рівня, що підтримує в межах об'єктно-орієнтованої парадигми віддалені виклики методів та розподілені транзакції. Це компонентне середовище для

компонентів Enterprise JavaBeans, яке підтримує автоматичну синхронізацію Java об'єктів з базою даних.

EJB-контейнер підтримує такі базові служби при роботі з компонентами EJB:

– автоматична підтримка звертань до компонентів, розміщених на різних машинах;

– автоматична підтримка транзакцій;

– автоматична синхронізація стану баз даних та відповідних компонентів EJB у обидва боки;

– автоматична підтримка захищеності за рахунок аутентифікації користувачів, перевірки прав користувачів або компонентів на виконання викликаних ними операцій та авторизації відповідних дій;

– автоматичне керування життєвим циклом компонента (послідовністю переходів між станами «немає» – «ініціалізований» – «активний») і набором компонентів аналогічно до ресурсів, тобто видалення компонентів, що стали непотрібними; завантаження нових компонентів; балансування навантаження між наявними компонентами.

WEB-контейнер. Компонентним середовищем для роботи Web-компонентів є Web-контейнер, що постачається в межах будь-якої реалізації платформи J2EE. Web-контейнер реалізує такі служби, як керування життєвим циклом компонентів і набором компонентів аналогічно до ресурсу, тобто розпаралелювання незалежних робіт, виконання віддалених звертань до компонентів, підтримка захищеності за допомогою перевірки прав компонентів і користувачів на виконання різних операцій.

Результати роботи компонентів EJB перетворюються Web-компонентами в динамічно генеровані HTML-сторінки, і надсилаються назад до користувача, постаючи перед ним у вікні браузера.

Роль Web- і EJB-контейнерів. Процеси та синхронізація. Розбиває прикладну програму J2EE на низку процесів із потоками керування, які взаємодіють, Web- або EJB-контейнер автоматично. На їх роботу можна впливати, конфігуруючи як J2EE-сервер у цілому, так і конкретні прикладні програми. Усі методи допоміжних класів, які використовують Web-компоненти й компоненти EJB, слід оголошувати синхронізованими. Компоненти J2EE, що

працюють у межах контейнерів, можуть створювати власні окремі потоки, але робити це потрібно з великою обережністю, оскільки цими потоками контейнер керувати не зможе, і вони можуть зашкодити роботі інших компонентів.

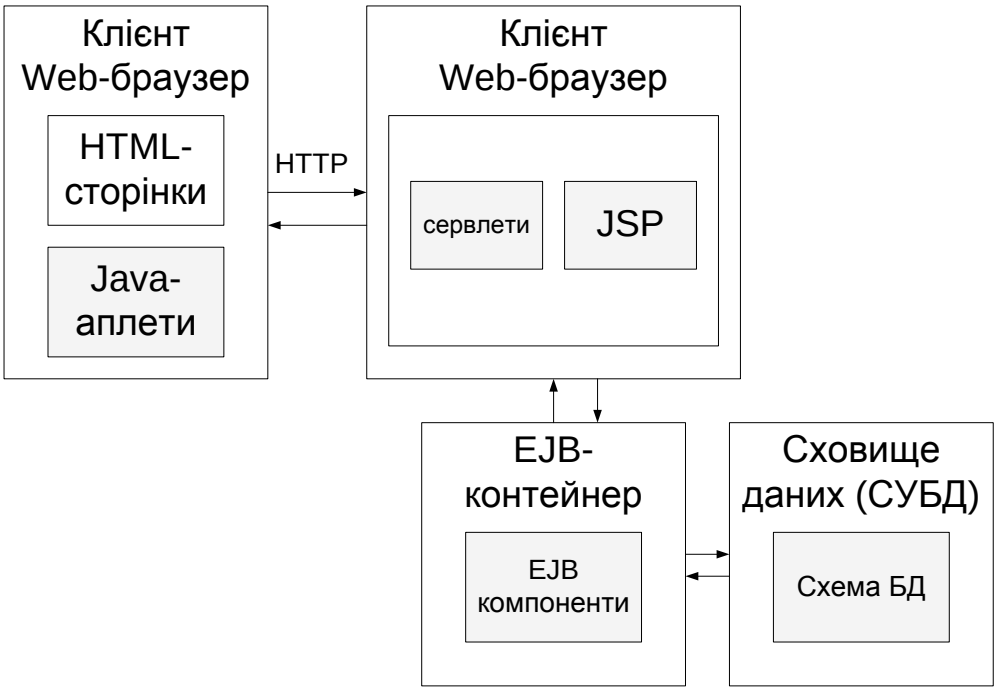

Рис. 2.60. Архітектура прикладної програми J2EE

Деякі сценарії прикладних програм та роль Web- і EJB-контейнерів. Нижче розглянуто лише кілька сценаріїв прикладних програм. Специфікація J2EE і відповідні технології мають тенденцію приймати і підтримувати різноманітність, а сценарії прикладних програм визначають, які точно APIs збираються використовуватися, щоб надати функціональність рівня прикладного програмного забезпечення. Вибір рівня реалізації прикладної програми – вибір оптимального рішення між функціональним різноманіттям і складністю. J2EE-модель програмування потрібна, щоб розглянути сценарії прикладних програм, які використовують Web-контейнер та EJB-контейнер як додаткові логічні сутності. Деякі ключові сценарії, включаючи ті, в яких Web-контейнера або EJB- контейнера, а можливо й обох, немає, відображено на рис. 2.61. Проста прикладна програма відображає свою багаторівневу модель. Це визначення передбачає наявність як Web-контейнера, так і EJB-контейнера.

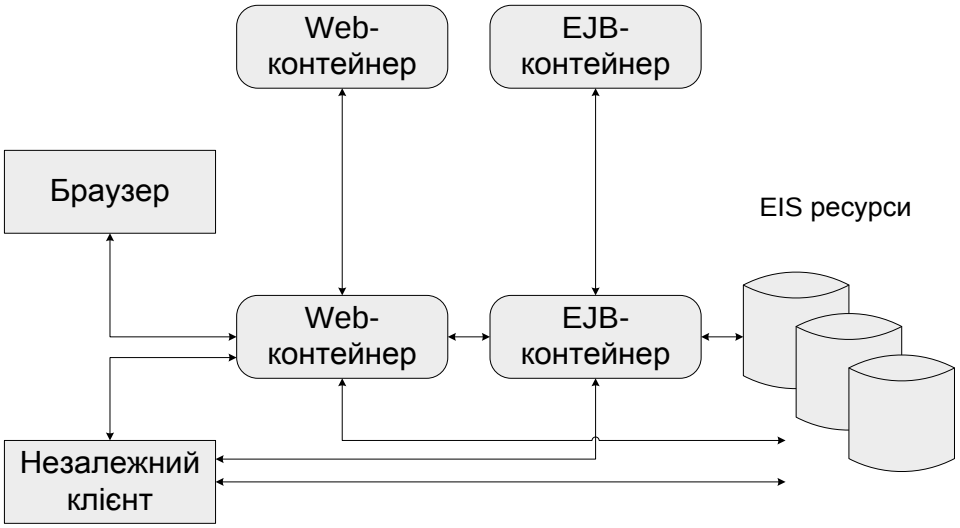

Рис.2.61. Сценарії

На вибір сценарію створення прикладного програмного забезпечення суттєво впливають такі корпоративні вимоги:

– необхідність швидко й часто змінювати вигляд прикладної програми;

– необхідність поділяти прикладні програми на представлення (інтерфейс користувача) та бізнес-логіку, щоб збільшити модульність;

– необхідність спрощувати процес подання інформації користувачам, які виконують завдання разом, причому робота кожним з них має виконуватися порівняно незалежно, але пов’язаними сценаріями;

– необхідність мати розробників, які знаються на офісних прикладних програмах, звільнених від розробки GUI і дизайну, в яких вони не можуть бути високо кваліфікованими;

– необхідність мати певний словник, щоб передати бізнес-логіку командам розробників, які розуміють вплив людського фактора й естетики прикладного програмного забезпечення;

– здатність створювати офісні прикладні програми, використовуючи компоненти з різних джерел, включаючи наявні компоненти бізнес-логіки;

– здатність розгортати виконувані (transactional) компоненти через численні апаратні та програмні платформи, незалежно від основної технології бази даних;

– здатність перетворювати внутрішні дані на зовнішні без додаткової інформації про споживача даних і виконувати це слабкозв'язаним способом.

Безсумнівно, врахування не в повному обсязі деяких або всіх названих вимог може впливати на рішення щодо проектування рівнів прикладної програми. Модель програмування J2EE вирішує проблему шляхом створення трирівневої прикладної програми так, щоб у подальшому перехід на багаторівневу архітектуру спрощувався за рахунок повторно використовуваних компонентів. Хоча розумно говорити про «недовговічність» логіки представлення (інтерфейс користувача часто змінюють), все ж таки існує порівняно мало змінюваний інтерфейс між рівнями прикладного програмного забезпечення, який зв'язує інтерфейс користувача з бізнес-логікою. Такий підхід широко використовують у разі організації взаємодії схем баз даних і даних. Нині за рахунок використання єдиного середовища доступу до ресурсів (EIS) з'являється можливість частої зміни коду прикладної програми, тобто довговічність прикладної програми значно знижується. Таким чином, модель програмування J2EE прискорює розвиток, сприяє повторному використанню компонентно-орієнтованого коду та є важелем для посилення міжрівневої взаємодії, яка використовує інтеграцію рівнів у моделі програмування J2EE.

Безліч сценаріїв функціонування прикладних програм, які продукт J2EE здатний підтримувати, ілюструє рис. 2.61, причому немає повної переваги одного сценарію над другим. У той же час продукт J2EE дозволяє підтримувати деякі або всі сценарії, що є перевагою, враховуючи те, що сценарії індивідуальні та розробляються з використанням технологій та протоколів, актуальних для розробника програми.

Багаторівневий сценарій програми. Сценарій програми, в якому Web-контейнер містить Web-компоненти, які майже повністю обробляють логіку представлення цієї прикладної програми, ілюструє рис. 2.62. Передача динамічного Web-вмісту клієнта є завданням JSP-сторінок (підтримується сервлетами). EJB-контейнер містить компоненти, які, з одного боку, відповідають на запити з Web-рівня, а з другого, мають доступ до ресурсів EIS.

Здатність ізолювати дані, що йдуть від кінцевого користувача,– перевага такого сценарію, оскільки прикладна програма неявно масштабована, але більш важливо, що офісна функціональність програми деякою мірою ізолюється від «look and feel» кінцевого користувача. Такий підхід має сенс, враховуючи що XML включений як невід'ємна частина цього сценарію.

Рис. 2.62. Багаторівневий сценарій функціонування програмного забезпечення

Здатність одночасно генерувати й опрацьовувати дані у форматі XML у Web-контейнері можна розглядати як дуже гнучку взаємодію з різними клієнтськими платформами, складність яких може бути різною, починаючи від універсального XML-підтримувального браузера закінчуючи спеціалізованою XML-генерувальної машини, орієнтованої на вертикальні рішення.

Незалежно від сфери застосування XML-дані передають через протокол HTTP. Терміном «передача даних XML» позначають модель програмування, в якій XML використовують для обміну інформацією замість застосування для цього об’єктної моделі, протилежної об’єктній моделі Java. Отже, XML можна розглядати як доповнення до мови Java.

На Web-рівні часто виникає запитання, що використовувати: JSP-сторінки або сервлети. Модель програмування J2EE використовує JSP-технологію, як основний засіб програмування у Web-контейнері. JSP-сторінки залежать від функціональності сервлетів, але модель програмування J2EE визначає JSP-сторінки як найбільш ефективний інструмент програмування для Web-інженерів, тому в Web-контейнері, спроектованому для ство-

рення динамічного вмісту для Web-клієнтів, використання технології JSP слід розглядати як норму, а використання сервлетів як виняток.

Сценарій незалежного клієнта. Сценарій, який використовує незалежного клієнта у процесі функціонування прикладної програми, показано на рис. 2.63.

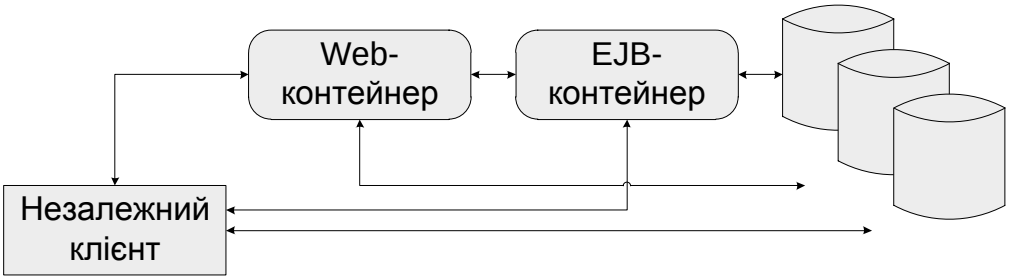

Рис. 2.63. Сценарій з використанням незалежного клієнта

У моделі програмування J2EE розглядають три типи незалежних клієнтів:

1. EJB-клієнти, які безпосередньо взаємодіють з EJB-сервером, причому корпоративні біни містяться в EJB-контейнері. Такий сценарій показано на рис. 2.64, у цьому сценарії допускається використання RMI-IIOP і EJB-сервера, який буде звертатися до ресурсів EIS за допомогою JDBC (або конекторів), показано на рис. 2.64

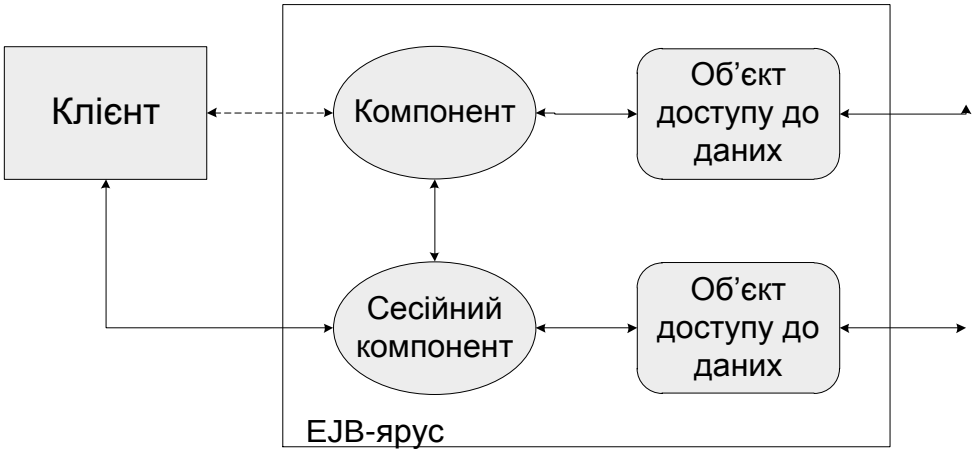

Рис. 2.64. EJB-клієнти безпосередньо взаємодіють з EJB-сервером

2. Незалежні Java-клієнти, які звертаються до ресурсів EIS безпосередньо, використовуючи JDBC або конектори. У цьому сценарії логіка представлення та бізнес-логіка за призначенням розміщені на клієнтській платформі й можуть бути інтегровані в одну прикладну програму. Цей сценарій перемі-

щує середній рівень на платформу клієнта й, по суті, є клієнт-серверним сценарієм програми, з усіма притаманними йому проблемами поширення, підтримки та оновлення.

3. VB-клієнти, які використовують динамічний Web-контент, що подається найчастіше у вигляді XML-даних. У цьому сценарії Web-контейнер керує XML-перетвореннями й забезпечує Web-зв'язок з клієнтами. Логіка представлення передається для керування на клієнтський рівень, а Web-рівень керує бізнес-логікою і прямим доступом до ресурсів EIS. В ідеалі, бізнес-логіка виноситься на EJB-сервер, де розвинена компонентна модель може її значно підсилити.

Сценарій трирівневої Web-прикладної програми. EJB-сервер є достатньо потужним інструментом для створення розподіленого програмного забезпечення. Специфікація J2EE не зобов'язує використовувати дво-, три- або багаторівневі моделі прикладних програм, тобто потрібно використовувати інструменти, які відповідають складності задачі. Сценарій трирівневої Web-прикладної програми нині дуже поширений; Web-контейнер фактично містить логіку представлення і бізнес-логіку, а за допомогою JDBC можна отримати доступ до ресурсів EIS. Сценарій функціонування трирівневої Web-прикладної програми подано на рис. 2.65.

Рис. 2.65. Сценарій трирівневої Web-прикладної програми

Використання Web-контейнера у сценарії функціонування Web-орієнтованих прикладних програм подано на рис. 2.66, у якому, варто враховувати, що термін «Web-контейнер» використано в дуже вузькому значенні.

Наприклад, якщо такий продукт J2EE вибрано для реалізації, то у Web-прикладній програмі Web-контейнер і EJB-контейнер розміщені разом (тобто міжконтейнерна взаємодія оптимізована та деталі реалізації закриті), модель програмування J2EE використовує компоненти, встановлені на такій самій платформі, як за багаторівневим сценарієм.

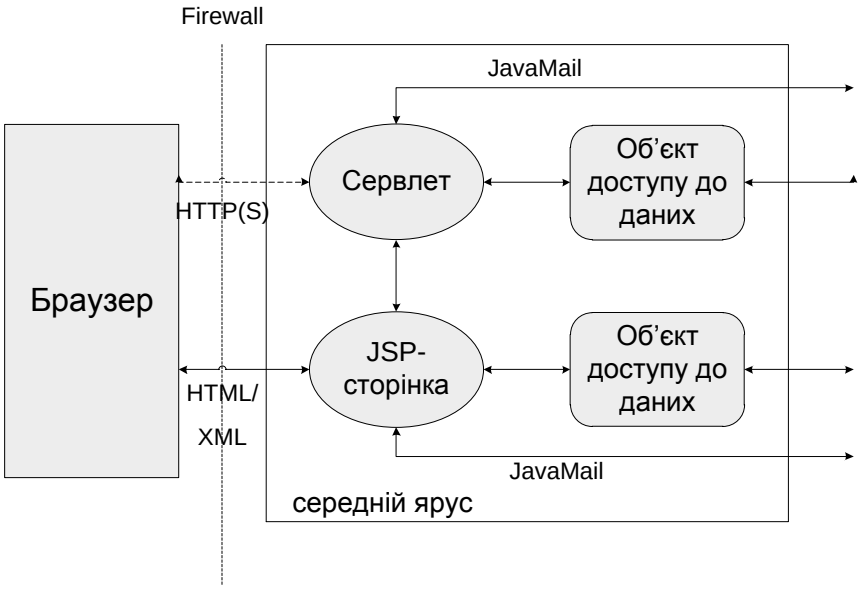

Рис. 2.66. Web-контейнер у сценарії функціонування Web-прикладних програм

Сценарій «Бізнес-Бізнес». Сценарій взаємодії бізнес-процесів у процесі функціонування Web-орієнтованого програмного забезпечення, який називають «Бізнес-Бізнес», подано на рис. 2.67.

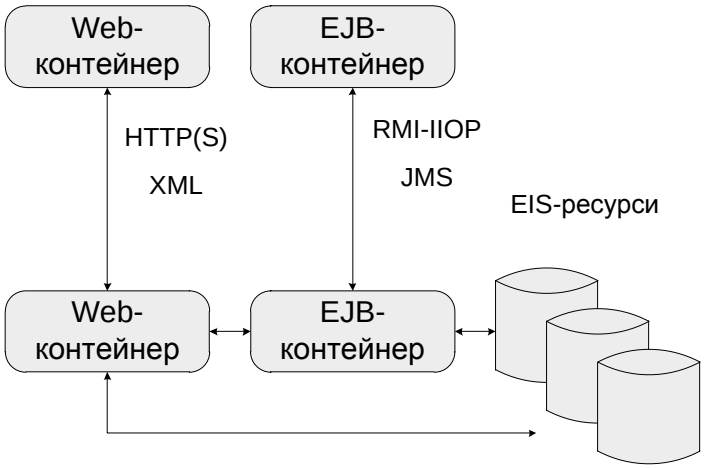

Рис. 2.67. Сценарій Бізнес-Бізнес

Цей сценарій акцентує увагу на міжрівневій взаємодії між Web- і EJB-контейнерами. Модель програмування J2EE пропонує використовувати XML-дані як головний засіб комунікації між Web-контейнерами, що зручно у разі розробки та розгортання комерційних рішень, в основу яких покладено Web. Міжрівневі комунікації між EJB-контейнерами в даний час найкраще рішення для Internet-середовища.

2.9. Висновки

1. Для створення відкритої розподіленої системи необхідне використання загальноприйнятих мов опису інтерфейсу програмної компоненти.

2. На сьогодні наявна низка апробованих на практиці стандартів для передачі даних у гетерогенних розподілених системах: XML, XSD, SOAP і WSDL.

3. Використання стандартів дозволяє створювати розподілені системи, не прив'язані жорстко до засобу розробки програм або транспортного протоколу.

4. Відкритий характер специфікації SOAP допускає як реалізацію деякої додаткової функціональності, що застосовується у проміжному середовищі, так і прийняття нових стандартів і розширень, які використовують заголовки SOAP.

5. Разом з тим, це може зумовити певні труднощі у разі взаємодії, заснованих на WSDL і SOAP програмних компонентів різних розробників.

6. Платформи розглянуто з погляду побудови на них складних гетерогенних Web-систем, тому деякі з популярних технологій не наведено в детальному огляді через неможливість або недоцільність їх використання як базової платформи. З розглянутого можна виокремити такі основні підходи до побудови архітектури серверних прикладних програм:

окреме виконання запитів, коли під час кожного запиту динамічного вмісту запускається окрема програма для обробки запитів, яка генерує вміст, що передається клієнтові. Цей підхід використовують у класичних CGI-скриптах;

накопичення виконуваних процесів, підхід аналогічний попередньому, але, якщо запит виконується повторно, то заново програма не запускається, а обробка передається існуючому процесу. Такий підхід застосовують у технологіях Java Servlets, Fast CGI;

шаблони сторінок, коли під час запиту шаблони заповнюють динамічним вмістом, який інтерпретується мовою сценаріїв, але не завжди створений у обчислювальному процесі. Підхід застосовують у технологіях ASP, JSP, PHP;

розширення Web-сервера, коли Web-сервер використовує особливі розширення для обробки динамічного змісту, специфічні для Web-сервера. Цей підхід використовують в IS API, NSAPI, mod_perl.

7. Rich Internet Application – це прикладна програма, доступна через Internet, збагачена функціональністю традиційних настільних прикладних програм, не підтримується браузерами безпосередньо. У RIA значна частина функціонала винесена у клієнтську частину, тобто виконується на стороні клієнта.

8. Основні механізми дескрипторів розгортання Web-орієнтованих прикладних програм на платформі J2EE реалізують такі компоненти платформи як EJB-контейнер та Web-контейнер. Наявні безліч способів, якими можливо реалізовувати специфічний набір бізнес-вимог, логічну модель і описувати рішення, що задовольняють поставленим вимогам. Ураховуючи ці вимоги, складають різноманітні сценарії функціонування прикладних програм, серед яких можна назвати сценарій трирівневої Web-прикладної програми, який дозволяє не застосовувати EJB-контейнер, тобто використовувати менше ресурсів та витрачати менше часу і процесорної пам’яті.

2.10. Запитання для самоконтролю

1. У чому полягають відмінності між мультипроцесорами та мультикомп’ютерами?
2. Коротко охарактеризуйте гомогенні та гетерогенні комп’ютерні розподілені системи.
3. На які категорії поділяють операційні системи для розподілених комп’ютерів? Коротко охарактеризуйте кожну з них.
4. Що являє собою проміжне середовище?
5. Назвіть основні служби платформи розподілу.
6. Дайте визначення поняття «розподілене середовище».