3. Захист даних, переданих між компонентами системи, від перегляду і змінювання третіми сторонами потребує, щоб передані між компонентами повідомлення візувати електронним підписом і шифрувалися як клієнтом, так і сервером.

Функції безпеки можна забезпечувати транспортним протоколом, який використовується проміжним середовищем, самим середовищем, або ними обома одночасно.

2.7. Опис інтерфейсу програмної компоненти

2.7.1. Мова і схеми XML (Extensible Markup Language)

Мова XML нині набула низки різноманітних застосувань і є основою для великої кількості загальноприйнятих специфікацій, які використовують у розподілених системах (зокрема у мовах XML, XSD, SOAP, WSDL), та основні з яких подано на рис. 2.48.

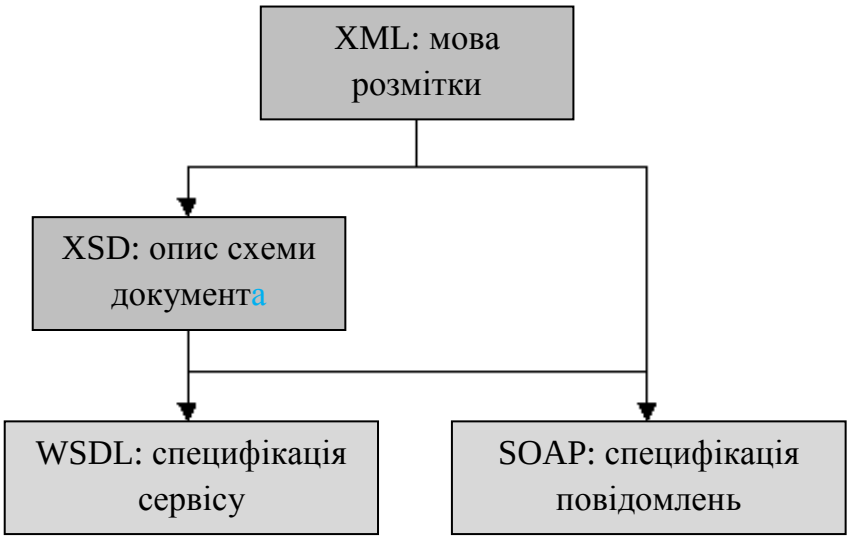

Рис. 2.48. Специфікації, які мають в основі XML-формат

Мова XML – це мова розмітки текстового документа, поданого у вигляді сукупності іменованих, деревоподібних вкладених елементів, кожний з яких може мати деяке текстове значення й набір атрибутів, у яких є ім’я і просте значення (рядок). Мова XML є абстрактною мовою розмітки, яка не визначає ніякого змісту елементів документа. Документи XML добре читає як людина, так і численні програмні аналізатори. У разі природного підходу до

імен елементів і атрибутів вона є мовою, яка самодокументується. Перед деревоподібною структурою елементів, що мають єдиний корінь, можуть перебувати окремі елементи з метаінформацією, яка свідчить, зокрема, про кодування документа й версію мови.

Основними недоліками XML з погляду обміну повідомленнями – є незручне, через його деревоподібну структуру, подання відносин «багато до багатьох», а також більші витрати часу на передачу й обробку повідомлень мовою XML порівняно з двійковим представленням аналогічних даних.

Оскільки властиве XML відкрите подання інформації не завжди зручне з погляду безпеки, то наявні специфікації XML-DigitalSignature і XML-Encrypton, призначені для передачі в XML-форматі конфіденційної інформації. Перша специфікація дозволяє додати до XML-документа цифровий підпис, друга – зашифрувати XML-документ або окремі його елементи.

Однією з переваг XML є наявність мов специфікацій, що визначають правильний XML-документ. Спочатку цю функцію виконував DTD (Document Type Definition), однак нині загальноприйнятим стандартом є специфікація схем XML (XML Schema Definition, XSD). Файл з описом схеми XML визначає такі параметри: словник документа (імена елементів і атрибутів); синтаксис коректного документа; складні типи даних.

2.7.2. SOAP: мова повідомлень розподіленої системи

Стандартизація опису мови XML дала широкі можливості для побудови на його основі мов опису повідомлень, переданих між програмними компонентами, і мов опису сервісів програмних компонент. Наприкінці 90-х років розпочали розробляти дві специфікації для побудови розподілених гетерогенних систем – SOAP і XML RPC. Специфікація XML RPC підтримується нині більшою кількістю мов, але має менше можливостей і не підтримується стандартною бібліотекою .NET Framework.

Оскільки в момент розробки таких специфікацій протокол HTTP був найпоширенішим у міжмережних екранах, то його було обрано як стандартний транспортний протокол для створення гетерогенних проміжних середовищ. Хоча специфікація SOAP не прив’язана жорстко до якого-небудь транспортного

протоколу, який використовує SOAP і WSDL, проміжне середовище названо **web-службою** (web services). Web-сервіси використовують дві мови прикладних програм – мову опису повідомлень SOAP і мову опису сервісів й інтерфейсів WSDL.

Рекомендацію SOAP спочатку розробляли як специфікацію для віддаленого виклику методів і розшифровували як Simple Object Access Protocol. Повідомлення SOAP є XML-документом, який називають **конвертом** або **пакетом** (envelope). Цей документ містить заголовки з метаінформацією в елементі soap:Header і тіло повідомлення в елементі soap:Body. У заголовках пакета міститься інформація прикладної програми, яка може використовуватися проміжним середовищем. Завдяки тому, що основний стандарт не обмежує змісту заголовків, SOAP є розширюваною специфікацією, і нині триває процес стандартизації її розширень.

Через різні причини нині розрізняють два різні способи представлення інформації в тілі пакета SOAP: кодування SOAP RPC (у двох варіантах) і кодування SOAP Document. Кодування SOAP RPC призначено винятково для передачі параметрів віддаленого виклику й визначає повідомлення як ім’я методу і список параметрів. У разі використання кодування SOAP Document, що є нині фактичним стандартом, повідомлення являє собою XML-документ, який має схему і простір імен, які задано в описі сервісу мовою WSDL (Web Service Definition Language). Зазвичай повідомлення складається з імені методу віддаленого об’єкта і списку його параметрів, сама специфікація кодування ніяк не фіксує його змісту.

Опис типів переданих даних за специфікацією SOAP Document містить схему XML, яка визначає коректні повідомлення,що отримуються програмним компонентом у тілі пакета SOAP, та включає такі параметри:

– опис вхідних і вихідних повідомлень, які зв’язуються з описаними типами даних;

– опис операцій (сервісів програмної компоненти), з кожною з яких зв’язується вхідне й вихідне повідомлення;

– опис типів портів (ідентифікаторів програмних компонент), з кожним з яких зв’язується деякий набір операцій;

– опис прив’язок (binding), що зв’язують типи портів і їх повідомлень з певним типом кодування тіла пакета, а також із версією протоколу SOAP;

– опис портів, що зв’язують типи портів і відповідні прив’язки з конкретними URL;

– загальний опис служби (інтерфейсу програмної компоненти) як сукупності портів.

2.7.3. WSDL: опис інтерфейсу програмної компоненти

Для опису інтерфейсу програмної компоненти, включаючи специфікацію коректних повідомлень, було розроблено мову WSDL. Опис мовою WSDL містить сім складових (рис. 2.49).

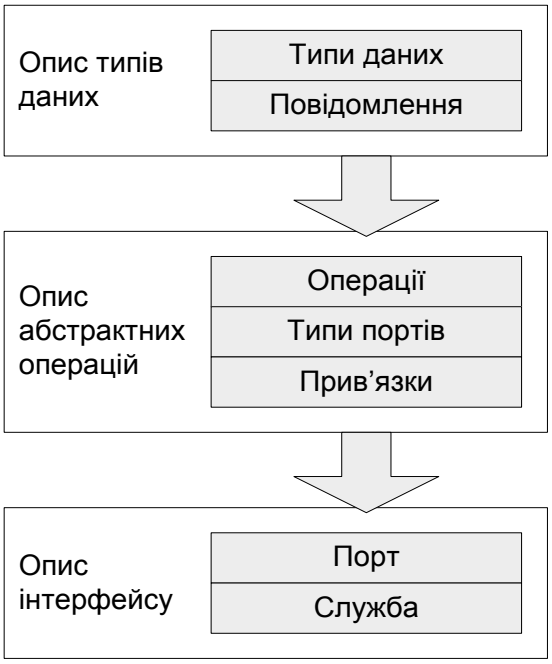

Рис. 2.49. Складові WSDL-документа

2.7.4. Серіалізація об’єктів .NET Framework

На відміну від прикладних програм на некерованому коді, прикладна програма .NET Framework не обов’язково виконується у вигляді окремих процесів, а може перебувати в межах одного процесу операційної системи у власних областях – доменах прикладної програми, які можна розглядати як деякі логічні процеси віртуальної машини Common Language Runtime (CLR).

Використання керованого коду за таких умов дозволяє гарантувати ізоляцію прикладної програми у межах своїх областей. У процесі передачі повідомлення між доменами прикладної програми деякого об’єкта для його класу слід визначати такі параметри: процедуру серіалізації, що дозволяє зберегти стан об’єкта в деякому зовнішньому сховищі (наприклад, у файлі або в повідомленні транспортного протоколу) за допомогою потоків введення-виведення; процедуру десеріалізації, що створює копію об’єкта за збереженим станом (рис. 2.50).

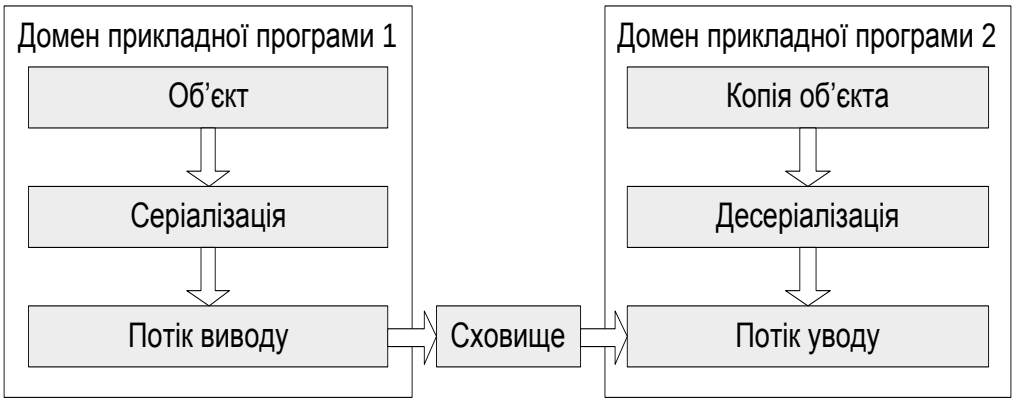

Рис. 2.50. Серіалізація й десеріалізація об’єкта

Слід зазначити, що загалом такі об’єкти можуть бути об’єктами різних класів і навіть створеними в різних системах розробки прикладних програм. Завдання серіалізації об’єкта, який містить лише поля з елементарних типів значень (спадкоємців класу *System.ValueType*) і рядків, не становить принципових труднощів. Для такого об’єкта у процесі серіалізації в потік записують самі значення всіх полів об’єкта, однак загалом об’єкт містить посилання на інші об’єкти, які, у свою чергу, можуть посилатися один на одного, утворюючи граф об’єктів (*object graph*). Самі посилання не можуть зберігатися в потоці введення-виведення, тому основне питання серіалізації – у який спосіб заміняти посилання.

Граф об’єктів – орієнтований граф $G = < V, E >$, у якому вершини – це об’єкти (множина V), а ребра направлені від об’єктів, що містять посилання, до об’єктів, на які посилаються (рис. 2.51). Усі об’єкти, які розглядаються у процесах серіалізації або десеріалізації описують у термінах об’єктно-орієнтованого підходу. В такому разі об’єкти, властивості яких наслідують інші об’єкти, є батьками, а об’єкти, які наслідують властивості батьків, є дітьми.

Наявність посилання на об’єкт *B* у об’єкті *A* є приналежністю пари <*A*,*B*> множині *E*. У графі всі поля об’єктів, які належать до простих типів значень і рядків, можна вилучити з розгляду в силу тривіальності їх серіалізації. Хоча формально рядки є посилальними типами, обмеженням реалізації рядків у CLI є те, що немає можливості змінити вже створений рядок, але є можливість тривіально обробляти рядки під час серіалізації, зберігаючи в потоці їх вміст.

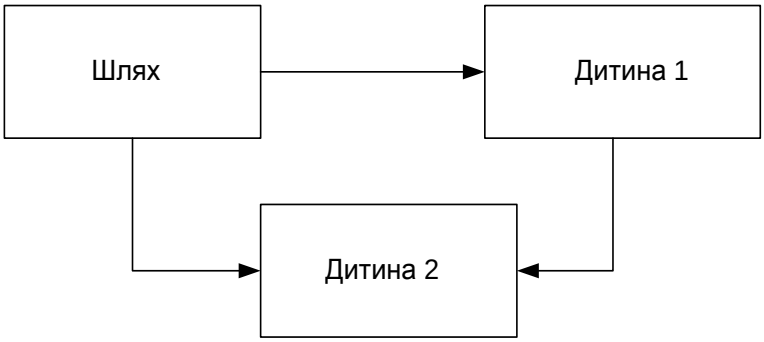

Рис. 2.51. Граф об’єктів

Існує окремий випадок, коли серіалізація виконується тривіально, у такому разі граф повинен мати вигляд орієнтованого дерева. У кожну вершину дерева, відмінну від кореня, направлене єдине ребро, існує єдина вершина, у яку не направлене жодне ребро, – корінь. За таких умов серіалізація може виконуватися від кореня методом у глибину, серіалізація полів-посилань функціонує аналогічно до серіалізації полів-значень. Якщо виявлено посилання на об’єкт, то замість нього у сховище розміщують значення полів об’єкта, на який посилаються, і деяку інформацію прикладної програми щодо типу об’єкта. Наприклад, якщо наявні ребра $<A, B_1>$, ... $<A, B_n>$, то функцію серіалізації *S* для об’єкта *A* можна подати як $S(A) = <V(A), <S(B_1), ... S(B_n)>>$, де функція *V* – значення полів-значень і рядків цього об’єкта.

Проблема серіалізації графу об’єктів полягає в тому, що посилання на той самий об’єкт може бути значенням поля різних об’єктів (наявні ребра $<A_1, B>$ і $<A_2, B>$). Можлива також наявність циклів у вигляді $A \rightarrow ... \rightarrow A$.

У цьому разі у процесі серіалізації об’єктам мають відповідати деякі ідентифікатори, й у сховищі окремо зберігається список об’єктів, позначений ідентифікаторами, а під час серіалізації замість посилань записують ідентифікатори об’єктів, на які посилаються. Якщо A_1, ... A_n – всі вершини графа, то

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.