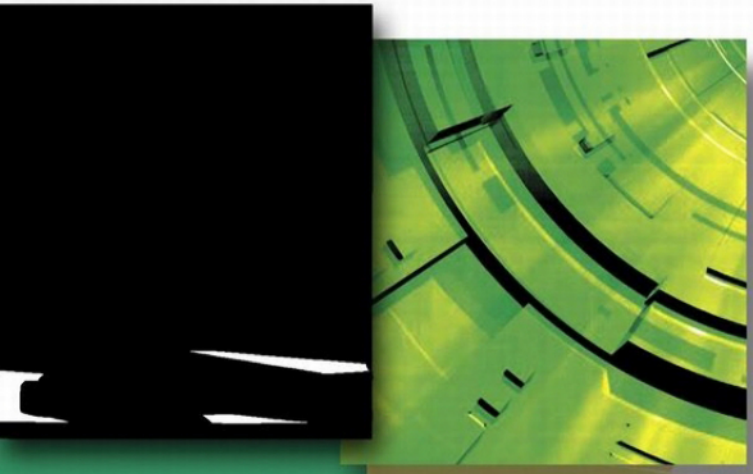
ТИМУР МАШНИН

bhv®

Web-сервисы Java



ОСНОВЫ ТЕХНОЛОГИИ WEB-СЕРВИСОВ В СПЕЦИФИКАЦИЯХ ПЕРВОГО И ВТОРОГО УРОВНЯ

СТАНДАРТЫ ТЕХНОЛОГИИ WEB-СЕРВИСОВ ПЛАТФОРМЫ JAVA

JAVA-СТЕКИ WEB-СЕРВИСОВ: Metro, CXF И Axis2

PRO

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

+

Материалы на www.bhv.ru

УДК 681.3.06
ББК 32.973.26-018.2
М38

Машнин Т. С.

М38 Web-сервисы Java. — СПб.: БХВ-Петербург, 2012. — 560 с.: ил. — (Профессиональное программирование)

ISBN 978-5-9775-0778-3

Рассмотрены основы технологии Web-сервисов в спецификациях первого и второго уровня, реализация технологии Web-сервисов в виде стандартов платформы Java и в таких распространенных Java-стеках Web-сервисов, как Metro, CXF и Axis2. Материал книги сопровождается более 70 примерами с подробным анализом исходных кодов. На сайте издательства находятся примеры проектов из книги, а также дополнительные материалы.

Для программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Марины Дамбиевой*
Зав. производством	*Николай Тверских*

Подписано в печать 31.10.11.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 45,15.
Тираж 1200 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0778-3

Оглавление

ГЛАВА 1

Архитектура XML Web-сервисов

Технология Web-сервисов обеспечивает взаимодействие между приложениями, работающими на различных платформах, с помощью программных компонентов — Web-сервисов.

Web-сервисы — это программные компоненты, которые имеют интерфейс, описанный в формате, пригодном для компьютерной обработки, как правило — это WSDL-формат, и взаимодействие с которыми осуществляется согласно их WSDL-описанию с помощью сообщений, передаваемых обычно по HTTP-протоколу.

Web-сервис является ресурсом, который характеризуется URI-адресом. При этом интерфейс Web-сервиса представляет его абстрактную функциональность, а конкретная реализация интерфейса Web-сервиса обеспечивает отправку и получение сообщений при взаимодействии клиента с Web-сервисом.

Различные реализации интерфейса Web-сервиса по-разному реализуют его общую функциональность, предоставляя многообразие услуг в рамках одного сервисного набора.

Конкретную реализацию интерфейса Web-сервиса обеспечивает *агент* — программный компонент, осуществляющий отправку и получение сообщений и принадлежащий *поставщику сервиса*. Клиент, осуществляющий запрос к Web-сервису с помощью клиентского приложения, также именуемого агентом, называется *потребителем сервиса*.

Как было сказано, взаимодействие с Web-сервисом осуществляется с помощью обмена сообщениями. Для того чтобы такой обмен был успешным, его механизм должен быть определен для потребителя сервиса его поставщиком. Такое определение механизма обмена сообщениями обеспечивается WSDL-описанием Web-сервиса.

WSDL-описание Web-сервиса определяет формат сообщений, участвующих в обмене, тип передаваемых данных, протокол передачи сообщений и адреса доставки сообщений Web-сервису.

Общепризнанным кроссплатформенным форматом обмена сообщений является XML-формат.

Архитектура XML Web-сервисов описывается набором моделей:

- Message Oriented Model — модель описывает сообщения Web-сервиса, включая структуру сообщений и механизм их доставки;
- Service Oriented Model — модель характеризует сервис, выполняемые действия при взаимодействии с сервисом, агентов поставщиков и потребителей сервиса, а также определяет метаданные, используемые для описания сервиса;
- Resource Oriented Model — модель описывает Web-сервисы как ресурсы, идентифицируемые URI-адресом, принадлежащие поставщикам сервиса и предоставляемые потребителям сервиса;
- Policy Model — модель описывает условия доступа к ресурсам.

Модель Message Oriented

Модель Message Oriented Model (MOM) описывает сообщения и их обработку, включая структуру сообщений, отношения между отправителем и получателем сообщений, а также механизм их передачи.

MOM определяет понятия адреса, политики доставки, сообщения, тела сообщения, корреляции сообщения, оболочки сообщения, модели обмена сообщениями Message Exchanage Pattern (MEP), заголовка сообщения, получателя сообщения, надежности сообщения, отправителя сообщения, последовательности сообщений, транспортировки сообщений.

Для того чтобы сообщение было доставлено получателю, необходим адрес доставки сообщения.

Формат адреса доставки сообщения зависит от механизма транспортировки сообщения. В случае HTTP-протокола передачи сообщений — это URL-адрес получателя.

Адрес доставки сообщения может содержаться в самом сообщении, как правило, в его оболочке.

Политика доставки — это условия доставки сообщения, которые могут заключаться в гарантиях доставки сообщения, его кодировке при передаче, формировании отчета о доставке и т. д.

Модель MOM определяет сообщение как единицу данных, передаваемую от одного агента другому. Структура сообщений определяется в WSDL-описании Web-сервиса.

Сообщение состоит из конверта (оболочки), одного или нескольких заголовков и тела сообщения. Конверт сообщения служит контейнером для его компонентов и может содержать информацию об адресе доставки сообщения. Заголовок сообщения содержит информацию о сообщении, относящуюся к системе безопасности, механизму транзакций, маршрутизации сообщения и т. д. Заголовок сообщения может обрабатываться независимо от самого сообщения, поэтому каждое сообщение может иметь различные заголовки, определяющие способы обработки тела со-

общения. Тело сообщения содержит его контент или адрес соответствующего ресурса данных.

Клиент может вызывать Web-сервис с помощью простых HTTP-запросов GET и POST. В случае GET-запроса заголовок сообщения — это HTTP-заголовок, а параметры URL — это тело сообщения. В случае POST-запроса заголовок сообщения — это HTTP-заголовок, а тело сообщения — это передаваемые данные.

В ответ на HTTP GET- и POST-запросы XML Web-сервис передает клиенту сообщения в XML-формате.

Если клиент взаимодействует с Web-сервисом по SOAP-протоколу, тогда сообщения, участвующие в обмене, имеют SOAP-структуру, состоящую из конверта, заголовков и тела.

При обмене сообщениями с Web-сервисом устанавливается соответствующий контекст, в котором сообщение запроса ассоциируется с сообщением ответа, что важно в случае асинхронного обмена.

Обмен сообщениями с Web-сервисом регламентируется шаблоном Message Exchange Pattern (MEP), который определяет поток сообщений (последовательность) между агентами, включая обработку ошибок и связи между входящими и выходящими сообщениями.

Шаблоны MEP могут описывать обмен сообщениями не только с одним Web-сервисом, но и последовательности, в которых Web-сервисы взаимодействуют друг с другом.

Модель MOM определяет надежность и гарантированность сообщения как степень уверенности, что сообщение будет доставлено в необходимом порядке. Механизм надежности и гарантированности уменьшает ошибки и обеспечивает информацию о статусе доставки, делая возможным, в случае возникновения ошибки доставки, повторную отправку сообщения и восстановления его порядка доставки.

Модель Service Oriented

Модель Service Oriented Model (SOM) описывает сервис и действия.

Модель SOM определяет сервис как ресурс, представляющий возможность выполнения задач, реализующих функциональность, которая определена поставщиком сервиса. Чтобы сервис мог использоваться, он должен быть реализован хотя бы одним агентом.

Web-сервис, так же как и Web-ресурс, идентифицируется URI-адресом, однако главное отличие Web-сервиса от Web-ресурса в том, что Web-сервис необязательно должен иметь представление — данные состояния ресурса в общедоступных форматах XML, HTML, CSS, JPEG, PNG, получаемые с помощью метода HTTP GET.

Web-сервис имеет свое описание, которое представляет собой набор документов, подлежащий компьютерной обработке и содержащий описание интерфейса Web-сервиса и его поведения. Описание Web-сервиса может быть реализовано с помощью языка Service Description Language (SDL) в виде набора XML-документов.

Описание Web-сервиса может использоваться для его конструирования, развертывания, определения местонахождения, установления связи между агентами потребителя и поставщика сервиса.

Web-сервис ассоциируется с действиями, решающими определенные задачи, которые представляют функциональность сервиса. Модель SOM определяет действие как операцию, которая может быть выполнена агентом в результате получения или отправки сообщения или другого изменения состояния. Обычно действия реализуются путем выполнения фрагмента программы, представляющей агента получателя или отправителя сообщения.

Агент — это программа, исполняющая действия и реализующая потребителя или поставщика Web-сервиса.

Последовательность и условия, при которых сложная система взаимодействующих между собой агентов обменивается сообщениями, описываются *хореографией Web-сервисов*. Хореография Web-сервисов — это модель, описывающая последовательность операций, состояний и условий, контролирующих взаимодействие сервисов. Результатом такого взаимодействия является выполнение определенной задачи, которую решает поставщик сервиса перед его потребителем.

Хореография Web-сервисов может описываться языком Choreography Description Language (CDL), который определяет композицию сервисов, их роли и связи, а также управление состоянием системы связанных сервисов.

Последовательность и условия, при которых один Web-сервис вызывает другие Web-сервисы для решения конкретной задачи, определяются оркестровкой Web-сервисов.

Модель Resource Oriented

Модель Resource Oriented Model (ROM) описывает ресурсы.

Модель ROM определяет ресурс как объект, имеющий имя-идентификатор, свое представление и собственника, обладающего правом назначать политику ресурса, которая определяет правила взаимодействия с ресурсом.

Web-сервисы являются разновидностью ресурсов, которые имеют описание, пригодное для компьютерной обработки и содержащее идентификатор ресурса в виде URI-адреса. С помощью описания потребитель узнает о полезности ресурса и устанавливает связь с ним. Описание облегчает поиск и доступ к ресурсу, включая в себя информацию о местонахождении ресурса, способах доступа к ресурсу и политиках ресурса.

Как было сказано ранее, Web-сервисы — это разновидность ресурсов, отличающаяся от обычных Web-ресурсов тем, что Web-сервисы необязательно должны иметь свое представление — данные состояния ресурса, получаемые с помощью запроса HTTP GET.

Поиск описаний Web-сервисов может осуществляться с использованием специальных сервисов, отвечающих за публикацию и поиск описаний согласно определен-

ным критериям. При статическом поиске описания Web-сервиса его потенциальный потребитель связывается с таким сервисом с помощью соответствующего программного обеспечения, например Web-браузера, и получает необходимое описание Web-сервиса. При динамическом поиске описания Web-сервиса агент потребителя сервиса напрямую обращается к специальному сервису для установки связи с агентом поставщика сервиса.

Передача описания Web-сервиса между поставщиком и потребителем сервиса может осуществляться и без специального сервиса — напрямую или с помощью других источников, таких как файловые системы, Web-сайты и др.

Модель Policy

Модель Policy Model описывает политики сервиса, представляющие собой ограничения на допустимые действия или состояния агентов или их собственников.

Политики могут определять ограничения относительно доступа к ресурсам/состояний ресурсов или возможных действий/состояний агентов поставщиков или потребителей сервисов.

Политики могут быть двух типов:

- политики разрешений;
- политики обязательств.

Политика разрешений позволяет агентам выполнять определенные действия, иметь доступ к определенным ресурсам, достигать определенного состояния.

За выполнением политики разрешений следит механизм защиты разрешений, гарантирующий соответствие использования сервиса — политики, установленной поставщиком сервиса.

Политика обязательств выдвигает требования для агентов исполнять определенные действия или достигать определенных состояний.

За выполнением политики обязательств следит механизм аудита, проверяющий выполнение обязательств, установленных поставщиком сервиса.

Политики могут иметь описание в формате, пригодном для компьютерной обработки. Описание политики определяет ее и используется для решения применения политики к конкретной ситуации.

Архитектура Service Oriented Architecture (SOA)

Web-сервисы являются программными компонентами распределенных приложений, имеющих сервис-ориентированную архитектуру SOA.

Распределенные системы могут создаваться с помощью технологии Web-сервисов, реализующей архитектуру SOA, или технологий COM/CORBA. Однако технологии COM/CORBA не обеспечивают в полной мере кроссплатформенность.

Реализация архитектуры SOA с помощью технологии Web-сервисов более приемлема для создания распределенных приложений, компоненты которых взаимодействуют через Интернет, разворачиваются и работают на различных платформах.

В архитектуре SOA распределенные системы состоят из взаимодействующих агентов поставщиков и потребителей сервисов, выполняющих определенные задачи. Агенты распределенной системы не работают в одной и той же среде выполнения и связываются друг с другом с помощью протоколов через сеть.

В распределенных системах скорость выполнения задач зависит от скорости удаленного доступа к агентам, а надежность — от ошибок коммуникации между агентами.

Архитектура SOA характеризуется следующими свойствами.

- *Логическое представление.* В логическом представлении SOA сервис — это абстрактное, логическое представление реальной программы, базы данных, бизнес-процесса и т. д., определенное в терминах функциональности.
- *Сообщения.* Сервис определяется в терминах сообщений, участвующих в обмене между агентами поставщиков и потребителей сервиса. Архитектура SOA не определяет внутреннюю структуру агентов, их реализацию, тем самым обеспечивая совместимость между любыми программными компонентами, удовлетворяющими требованиям обмена сообщениями в определении сервиса.
- *Описание.* Сервисы описываются метаданными, подлежащими компьютерной обработке. Описания сервисов обеспечивают публичность архитектуры SOA.
- *Степень детализации.* Сервисы используют небольшой набор операций с большим набором сообщений.
- *Передача по сети.* Взаимодействие с сервисами может осуществляться с помощью сетевого соединения.
- *Кроссплатформенность.* Сообщения, участвующие в обмене с сервисами, имеют стандартный кроссплатформенный XML-формат.

При реализации архитектуры SOA необходимо решать проблемы, связанные с ненадежностью транспортных протоколов, параллельным удаленным доступом, совместимостью программных компонентов, возможными ошибками одного или нескольких компонентов системы, обеспечением достаточной памяти для вызова объектов.

Архитектура SOA может быть реализована с помощью трех типов XML Web-сервисов:

- *RESTful Web-сервисы* обеспечивают обмен и управление представлениями ресурсов с помощью HTTP-методов GET, PUT, POST и DELETE;
- *RPC SOAP Web-сервисы* обеспечивают определенный набор операций;
- *XML Web-сервисы* обеспечивают получение, обработку и отправку сообщений.

Все три типа Web-сервисов имеют URI-идентификаторы и XML-формат сообщений. При этом RESTful и XML Web-сервисы могут использовать HTTP- или

SOAP/HTTP-протоколы для обмена сообщениями, а SOAP Web-сервисы могут использовать SOAP-протокол вместе с транспортными протоколами HTTP, SMTP, FTP и др.

Основные технологии архитектуры Web-сервисов

XML

XML-технология обеспечивает кроссплатформенный и расширяемый стандарт для формата данных.

При обсуждении XML-технологии мы будем опираться на спецификации XML 1.0, XML Information Set, Namespaces in XML 1.0 и XML Schema.

В архитектуре Web-сервисов XML-формат используется как для описания сервисов, так и для обмена сообщениями с Web-сервисами.

eXtensible Markup Language (XML) — это язык разметки документов, обеспечивающий текстовый формат хранения данных. Язык XML является подмножеством языка Standard Generalized Markup Language (SGML) и описывает определенный класс объектов, называемых *XML-документами*.

XML-формат представляет собой платформонезависимый способ структурирования информации путем представления документа в виде дерева элементов, каждый из которых может иметь набор атрибутов, представляющих пару "имя/значение", и содержать другие элементы и/или текст. При этом каждый элемент дерева может ссылаться на другие элементы с помощью своих атрибутов.

Приложения, работающие с XML-документами, для их обработки используют программный компонент — XML-процессор, обеспечивающий доступ к содержимому и структуре XML-документа. В технологии Java XML-процессоры представлены семейством Java for XML Processing (JAXP), включающим в себя процессоры DOM, SAX, StAX и TrAX.

XML-процессоры подразделяются на две категории — потоковые и объектные обработчики. При *потоковой обработке* XML-документа XML-процессор анализирует документ последовательно, в реальном времени, что экономит ресурсы памяти. SAX — это потоковый XML-процессор. При *объектной обработке* XML-документа XML-процессор создает в памяти объекты, представляющие состояние документа, что дает возможность в произвольном порядке анализировать его части. DOM — это процессор, поддерживающий объектную модель документа.

Процессор StAX является промежуточным между SAX и DOM, обеспечивая управление анализом XML-документа в приложении.

Процессор TrAX дает возможность трансформации XML-документа в другие форматы данных с использованием таблицы стилей XSLT. Существуют также XML-процессоры, трансформирующие XML-документы в формат HTML и XHTML с использованием каскадных таблиц стилей CSS.

Модель XML-документа описывает его в терминах логической и физической структуры. *Логическая структура* состоит из объявления, определения типа документа, элементов, комментариев, ссылок и инструкций по обработке документа.

Каждый XML-документ может начинаться с объявления, ограниченного тегами `<?xml` и `?>` и включающего в себя атрибуты `version` (версия XML-спецификации — на сегодняшний день только `1.0`), `encoding` (символьная кодировка документа, по умолчанию `Unicode`) и `standalone` (по умолчанию `no`, означает, что документ может иметь свое внешнее описание).

Далее в XML-документе может находиться описание структуры документа — определение типа документа DTD (Document Type Definition) или ссылка на внешний DTD-файл.

После объявления и определения типа документа его логическая структура представлена набором элементов, комментариев, ссылок и инструкций по обработке документа.

Элементы XML-документа, ограниченные тегами, которые называются *разметкой*, формируют его иерархическую структуру и могут иметь атрибуты, которые так же, как и элементы, способны хранить данные. XML-документ должен содержать как минимум один корневой элемент.

XML-теги определяют значение, типы данных, а не способ их отображения, как в формате HTML.

Атрибуты элементов представляют собой пары "имя/значение" и могут быть трех типов — строковый тип, лексемы и перечисления.

Дочерний элемент корневого элемента — это такое же его свойство, как и атрибут, за исключением того, что дочерний элемент может иметь свою сложную структуру.

Комментарии ограничиваются тегами `<!--` и `-->` и предназначены для документирования XML-документа.

Ссылки ограничиваются амперсандом (`&`) и точкой с запятой (`;`) и используются в XML-документах для подстановки при обработке документа вместо них символов или различного рода данных, описанных в определении DTD. Подставляемые данные могут быть обрабатываемыми XML-процессором, т. е. представлены в XML-формате, или необрабатываемыми — текстовые данные не формата XML, изображения и другие двоичные данные. Кроме того, для включения в XML-документ символьных данных, которые не следует обрабатывать XML-процессором, используется секция `CDATA`, ограниченная тегами `<![CDATA[` и `]]>`.

XML-ссылка — это ссылка на внешний объект, сущность, содержимое которого размещается в текущем месте, т. е. ссылка на сущность работает как подстановка и обеспечивает модульность XML-документа, в отличие от HTML-ссылки, являющейся инструкцией приложению для перехода на другой HTML-документ или фрагмент HTML-документа.

Инструкции по обработке ограничиваются тегами `<?` и `?>` и предназначены для передачи информации приложению, работающему с XML-документом.

Физическая структура XML-документа описывает его как набор сущностей, которые могут быть обрабатываемыми и необрабатываемыми, внешними и внутренними. XML-документ должен содержать как минимум одну сущность — корневую сущность документа. Сущности могут включаться в XML-документ с помощью ссылок.

XML-документы характеризуются условиями корректности (правильности) и действительности. *Правильный* XML-документ — это документ, имеющий один корневой элемент, элементы которого не перекрываются, удовлетворяющий другим требованиям спецификации XML. *Действительный* XML-документ — это документ, имеющий описание своей структуры и соответствующий этому описанию.

Помимо базового XML-синтаксиса, основные понятия XML-технологии, используемые в технологии Web-сервисов, — это XML Infoset, XML Schema и XML Namespaces.

XML Namespaces

Для предотвращения конфликта имен в XML-документах используются пространства имен, представляющие собой коллекции имен, в каждой из которых все имена уникальны, при этом каждая такая коллекция имеет свой уникальный идентификатор. Следовательно, каждое XML-имя может характеризоваться идентификатором *пространства имен* и *локальным именем* в пределах данного пространства имен.

Идентификатор XML-пространства имен является URI-адресом и используется в определении типа элемента XML-документа (имени начального и конечного тега элемента) и имени атрибута элемента.

Пространство имен объявляется с помощью зарезервированного имени `xmlns`, после которого может следовать префикс пространства имен, являющийся, по сути, сокращением идентификатора, и далее идет URI-идентификатор пространства имен.

Префикс пространства имен и локальное имя вместе составляют уточненное имя QName элементов и атрибутов XML-документа. Если пространство имен объявлено без префикса, тогда оно становится *пространством имен по умолчанию* для всех элементов XML-документа, не уточненных префиксом какого-либо пространства имен. Надо заметить, что это не относится к атрибутам XML-документа, которые для связывания с пространством имен всегда необходимо уточнять соответствующим префиксом. Поэтому корректный XML-документ не должен содержать разные по значению атрибуты с одинаковыми, не уточненными префиксом, локальными именами в пределах одного элемента.

Таким образом, механизм XML Namespaces позволяет присваивать уникальные имена элементам и атрибутам XML-документа, предоставляя каждому элементу и атрибуту свое уточненное QName-имя в пределах глобального XML-пространства имен `http://www.w3.org/XML/1998/namespace`.

XML Infoset

XML-формат делает возможным представление документа в виде иерархической структуры, определяющей его информационное пространство XML Infoset в виде множества информационных единиц, имеющих свойства.

Спецификация XML Information Set (XML Infoset) описывает абстрактную модель данных, представленных XML-документом, в виде набора информационных единиц. Каждая информационная единица является абстрактным описанием определенной части XML-документа и имеет набор связанных с ней свойств.

Необязательно каждый XML-документ должен иметь информационное пространство XML Infoset. Информационное пространство XML Infoset имеют лишь XML-документы, отвечающие правилам XML-синтаксиса согласно спецификации XML и соответствующие пространству имен согласно спецификации XML Namespaces.

Таким образом, правильно оформленный XML-документ — это сериализованная форма определенного информационного пространства.

Модель XML Infoset реализуется различными языками программирования в виде объектов, обеспечивающих доступ к частям XML-документа.

Спецификация XML Infoset определяет для XML-документа 11 типов информационных единиц — каждая с соответствующим набором свойств.

Информационное пространство XML-документа содержит как минимум корневую информационную единицу `document`. Все остальные информационные единицы информационного пространства XML-документа доступны с помощью свойств корневой информационной единицы `document` или свойств дочерних для `document` информационных единиц.

Информационная единица `document` имеет следующие свойства:

- `[children]` — список дочерних для `document` информационных единиц верхнего уровня, при этом список содержит как минимум одну информационную единицу — `document element`. Список также содержит информационные единицы `processing instruction` (инструкция по обработке информационной единицы) и `comment` (комментарий). Список может содержать информационный элемент `document type declaration` (DTD-описание или ссылка на DTD-описание структуры XML-документа);
- `[document element]` — идентификатор информационной единицы `document element`;
- `[notations]` — набор информационных единиц `notation` (нотации, которые дают возможность передавать приложениям различную информацию, например, способ обработки сущности, не подлежащей разбору);
- `[unparsed entities]` — набор информационных единиц `unparsed entity` (сущности, не подлежащие разбору);
- `[base URI]` — URI-идентификатор XML-документа;
- `[character encoding scheme]` — символьная кодировка XML-документа;

- [standalone] — если YES, тогда указывает автономность документа, т. е. возможность обработки XML-документа без привлечения других документов;
- [version] — версия XML-документа;
- [all declarations processed] — если true, тогда XML-процессор должен полностью обработать DTD-описание структуры XML-документа.

Каждый XML-документ состоит из элементов, ограниченных тегами и представляющих соответствующие информационные единицы element информационного пространства XML Infoset.

Информационные единицы element являются дочерними по отношению к информационной единице document element и имеют следующие свойства:

- [namespace name] — URI-идентификатор пространства имен, к которому принадлежит элемент;
- [local name] — локальная часть имени элемента, которая вместе со свойством [namespace name] образует имя элемента;
- [prefix] — префикс пространства имен элемента;
- [children] — список дочерних информационных единиц, содержащий информационные единицы element, а также информационные единицы processing instruction (инструкции для обработки), unexpanded entity reference (ссылка на необработанную внешнюю сущность, подлежащую разбору), character (единица текста — последовательности символов) и comment (комментарии) для данного элемента;
- [attributes] — набор информационных единиц attribute, представленный атрибутами элемента XML-документа;
- [namespace attributes] — набор информационных единиц attribute, представленный атрибутами элемента, объявляющими используемые элементом пространства имен;
- [in-scope namespaces] — набор информационных единиц namespace (пространства имен, используемые элементом);
- [base URI] — URI-идентификатор элемента;
- [parent] — информационная единица, являющаяся родительской.

Атрибуты элементов в виде пар "имя/значение" XML-документа представляют информационные единицы attribute информационного пространства XML Infoset. Атрибуты могут быть определены в XML-документе или установлены по умолчанию.

Атрибуты подразделяются на три типа — строковые, лексемы (знаковый тип) и перечисления.

Информационная единица attribute имеет следующие свойства:

- [namespace name] — URI-идентификатор пространства имен, к которому принадлежит атрибут;

- [local name] — локальная часть имени атрибута, которая вместе со свойством [namespace name] образует имя атрибута;
- [prefix] — префикс пространства имен атрибута;
- [normalized value] — нормализованное значение атрибута с преобразованием пробелов;
- [specified] — флаг, указывающий, что атрибут определен в начальном теге элемента или установлен по умолчанию в DTD-описании;
- [attribute type] — тип атрибута, объявленный в описании структуры XML-документа. Возможные значения — ID, IDREF, IDREFS (идентификаторы и ссылки на идентификаторы — знаковые типы), ENTITY, ENTITIES (имена сущностей, не подлежащих разбору — знаковые типы), NMTOKEN, NMTOKENS (лексемы имен — знаковые типы), NOTATION (информация, используемая приложением — строковый тип), CDATA (символьная строка — строковый тип) и ENUMERATION (список допустимых значений атрибута — перечисление);
- [references] — если тип атрибута равен IDREF, IDREFS, ENTITY, ENTITIES или NOTATION, тогда список информационных единиц element, unparsed entity или notation; если тип атрибута — ID, NMTOKEN, NMTOKENS, CDATA или ENUMERATION, тогда свойство не имеет значения;
- [owner element] — информационная единица element, к которой относится данная единица attribute.

Информационная единица processing instruction, представленная инструкцией обработки элемента в XML-документе, имеет следующие свойства:

- [target] — имя приложения, исполняющего инструкцию;
- [content] — данные для приложения в виде пар "имя/значение";
- [base URI] — URI-идентификатор инструкции;
- [notation] — информационная единица notation, определяющая имя приложения, которое исполняет инструкцию;
- [parent] — информационная единица, к которой относится данная инструкция.

Информационная единица unexpanded entity reference представлена ссылкой на необработанную XML-процессором внешнюю сущность, подлежащую разбору. Эта информационная единица имеет следующие свойства:

- [name] — имя сущности;
- [system identifier] — системный идентификатор сущности в виде URI-адреса сущности;
- [public identifier] — публичный идентификатор общедоступной сущности в виде URI-адреса сущности;
- [declaration base URI] — базовый URI-адрес, относительно которого разрешается системный идентификатор;
- [parent] — информационная единица element, к которой относится данная сущность.

Информационная единица `character` представлена символьными данными XML-документа и имеет следующие свойства:

- `[character code]` — символьная кодировка ISO 10646;
- `[element content whitespace]` — если `true`, тогда символ является пробелом;
- `[parent]` — информационная единица `element`, к которой относится данная информационная единица `character`.

Информационная единица `comment` представлена комментариями XML-документа и имеет следующие свойства:

- `[content]` — строка комментария;
- `[parent]` — информационная единица `document` или `element`, к которой относится данный комментарий.

Информационная единица `document type declaration` представлена ссылкой или самим DTD-описанием структуры XML-документа и имеет следующие свойства:

- `[system identifier]` — системный идентификатор внешнего DTD-описания в виде URI-адреса;
- `[public identifier]` — публичный идентификатор общедоступного внешнего DTD-описания в виде URI-адреса;
- `[children]` — список информационных единиц `processing instruction`, представленных инструкциями обработки DTD-описания;
- `[parent]` — информационная единица `document`, к которой относится данное DTD-описание.

Информационная единица `unparsed entity` представлена сущностью, не подлежащей обработке XML-процессором и объявленной в описании структуры XML-документа. Обычно это ресурс в двоичном формате или другом не XML-формате, например GIF, BMP и др. Информационная единица `unparsed entity` имеет свойства:

- `[name]` — имя сущности;
- `[system identifier]` — системный идентификатор сущности в виде URI-адреса сущности;
- `[public identifier]` — публичный идентификатор общедоступной сущности в виде URI-адреса сущности;
- `[declaration base URI]` — базовый URI-адрес, относительно которого разрешается системный идентификатор;
- `[notation name]` — имя нотации, определяющей формат сущности, которая не подлежит обработке;
- `[notation]` — информационная единица `notation` с именем, указанным в свойстве `[notation name]`.

Информационная единица `notation` представлена нотацией, объявленной в описании структуры XML-документа. Нотации используются для определения формата

сущностей (ресурсов), не подлежащих обработке XML-процессором, а также для указания приложений, получающих инструкции по обработке элементов XML-документа. Эта информационная единица имеет следующие свойства:

- `[name]` — имя нотации;
- `[system identifier]` — системный идентификатор нотации в виде URI-адреса;
- `[public identifier]` — публичный идентификатор общедоступной нотации в виде URI-адреса;
- `[declaration base URI]` — базовый URI-адрес, относительно которого разрешается системный идентификатор.

Информационная единица `namespace` представлена пространством имен элемента XML-документа и имеет следующие свойства:

- `[prefix]` — часть имени, следующая после `xmlns:`, если префикс отсутствует, тогда пространство имен является установленным по умолчанию;
- `[namespace name]` — пространство имен, следующее после префикса.

Как уже было сказано, XML-документы предназначены для хранения данных. Но приложение, использующее XML-документ, работает не непосредственно с документом, а с его информационным пространством XML Infoset, получаемым как результат разбора XML-документа XML-процессором. Однако информационное пространство XML Infoset может быть получено и другими способами, например, с использованием библиотек API DOM. В этом случае такое информационное пространство называется *синтетическим*.

XML-документы могут быть двух типов — документы, созданные с учетом правил, которым они должны подчиняться, и документы, не имеющие никаких правил.

При использовании XML-документа, не имеющего правил, вся ответственность за корректность написания документа лежит на его авторе.

Однако чаще всего XML-документы имеют описание своей структуры — XML-схему, представляющую собой набор логических и структурных правил. В этом случае проверку документа на соответствие правилам, определенным в описании структуры документа, производит XML-процессор.

XML-процессоры также могут быть двух типов — обработчики, которые обязаны осуществлять проверку XML-документа на соответствие описания его структуры, и обработчики, не осуществляющие такую проверку.

Описание структуры XML-документа, накладывающее ограничения на структуру и содержание документов данного типа, может быть создано с помощью различных языков, таких как Document Type Definition (DTD), XML Schema, RELAX NG и др.

Первым языком для создания XML-схем был язык Document Type Definition (DTD). Язык DTD является компактным и позволяет включать описание структуры XML-документа непосредственно в сам XML-документ. Но синтаксис языка DTD отличается от XML-синтаксиса, поэтому он прямо не поддерживается платформами и инструментами технологии XML. Кроме того, язык DTD не поддерживает пространства имен, а поддержка типов данных ограничена. Язык DTD не позволяет

определять элементы, содержащие целые и вещественные числа, даты и времена и др., и не может указывать сложные связи между элементами.

Язык XML Schema дает более богатые возможности по сравнению с DTD и устраняет его недостатки. Синтаксис XML Schema является XML-синтаксисом, кроме того, в языке XML Schema обеспечена поддержка пространств имен и всех необходимых типов данных. Также язык XML Schema позволяет создавать собственные типы данных и расширять существующие. Использование языка XML Schema обеспечивает трансформацию XML-документа в иерархию объектов определенных типов, доступных программным способом с помощью интерфейса (функциональность Post-Schema-Validation Infoset (PSVI)).

Язык RELAX NG, как и XML Schema, использует XML-синтаксис и по сравнению с XML Schema является более простым и легким в изучении. Язык RELAX NG сочетает в себе простоту DTD и богатые возможности XML Schema. Однако так как язык RELAX NG является в определенном смысле упрощением XML Schema, то он не поддерживает функциональность PSVI и имеет упрощенную типовую модель, что делает необходимым использование сторонних библиотек типов данных.

Здесь мы более подробно остановимся на языке XML Schema.

XML Schema

Язык XML Schema (XML Schema Definition (XSD)) позволяет очертить определенный круг XML-документов путем создания описания их структуры, накладывающим ограничения на данный тип документов и обеспечивающим их правильную интерпретацию. Описание XML-документов содержит определения элементов и атрибутов, которые могут появляться в данном типе документов, а также наследование элементов, включая порядок и количество потомков. Кроме того, язык XML Schema определяет тип содержимого элементов, типы данных элементов и атрибутов и, наконец, значения элементов и атрибутов по умолчанию и их фиксированные значения.

XML-документ ссылается на свое описание — XML-схему, созданную с помощью языка XML Schema, используя атрибуты *xsi*:schemaLocation и *xsi*:noNamespaceSchemaLocation, где *xsi* — префикс пространства имен, указанного в XML-документе с помощью атрибута xmlns:

```
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
```

Атрибут schemaLocation ссылается на XML-схему, имеющую определение пространства имен, а атрибут noNamespaceSchemaLocation — на XML-схему, не указывающую пространство имен для элементов и атрибутов. Атрибуты *xsi*:schemaLocation и *xsi*:noNamespaceSchemaLocation ссылаются на XML-схему своими значениями, представленными в виде URI-адреса схемы.

Корневым компонентом XML Schema является элемент <schema>, имеющий следующие атрибуты.

- Необязательный атрибут attributeFormDefault. Значение атрибута "qualified" указывает, что атрибуты XML-документа должны уточняться (квалифициро-

ваться) префиксом их пространства имен. По умолчанию значение атрибута — "unqualified".

- Необязательный атрибут blockDefault устанавливает по умолчанию значение атрибута block для всех элементов схемы. Возможные значения атрибута:
  - #all — в XML-документах запрещена замена базового элемента его производными, созданными любым способом;
  - extension — запрещена замена в XML-документах базового элемента производными элементами, полученными путем расширения базового типа элемента;
  - restriction — запрещена замена в XML-документах базового элемента производными элементами, полученными путем ограничения базового типа элемента;
  - substitution — в XML-документах запрещена любая замена базового элемента.
- Необязательный атрибут elementFormDefault. Значение атрибута "qualified" указывает, что элементы XML-документа должны уточняться (квалифицироваться) префиксом их пространства имен. По умолчанию значение атрибута — "unqualified".
- Необязательный атрибут finalDefault устанавливает по умолчанию значение атрибута final для всех типов данных схемы. Возможные значения атрибута:
  - #all — в XML-схеме запрещено создание производных типов любым способом;
  - extension — в XML-схеме запрещено создание производных типов путем расширения базового типа;
  - restriction — в XML-схеме запрещено создание производных типов путем ограничения базового типа;
  - list — в XML-схеме для простых типов запрещено создание производных типов путем создания списков;
  - union — в XML-схеме для простых типов запрещено создание производных типов путем объединения.
- Необязательный атрибут id — идентификатор схемы.
- Необязательный атрибут targetNamespace указывает пространство имен для элементов XML-документа, определяемых данной схемой.
- Необязательный атрибут version — версия схемы.
- Необязательный атрибут xml:lang определяет по умолчанию язык для всех комментариев схемы.

Элемент <schema> имеет также атрибут xmlns:xsd="http://www.w3.org/2001/XMLSchema", указывающий пространство имен для элементов и типов данных самой

схемы. При наличии префикса для данного пространства имен все элементы и типы данных схемы, включая элемент <schema>, должны уточняться этим префиксом (xsd:*schema*). Если префикс отсутствует, тогда атрибут xmlns указывает для схемы пространство имен по умолчанию. Данное пространство имен используется, чтобы очертить круг элементов и простых типов, относящихся к словарю языка XML Schema, а не к словарю автора схемы.

Элемент <schema> (рис. 1.1) может содержать вложенные элементы <include>, <import>, <redefine>, <annotation>, <type> (<simpleType>, <complexType>), <group>, <attributeGroup>, <element>, <attribute>, <notation>, <identityConstraint> (<key>, <unique>, <keyref>).

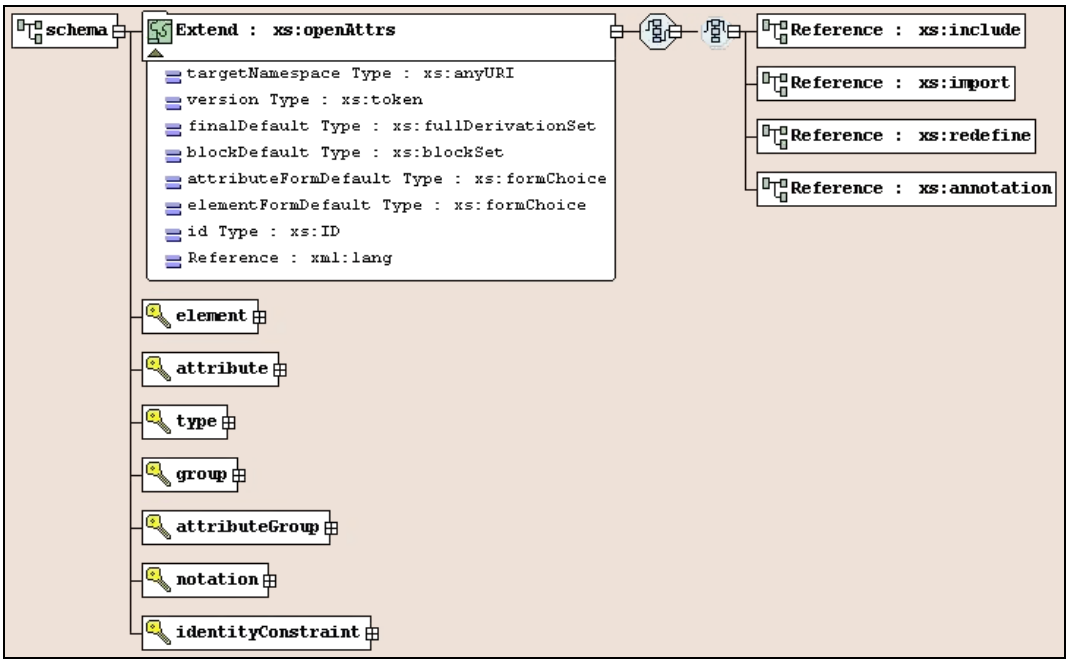


Рис. 1.1. Общая схема элемента <schema>

Язык XML Schema позволяет создавать описание структуры XML-документов, состоящее из нескольких схем, с помощью элементов <include> и <import>.

Элемент <include> добавляет все компоненты указанной схемы в основную схему, а элемент <import> дает возможность использовать компоненты указанной схемы в основной схеме.

Разница между элементами <include> и <import> состоит в том, что при использовании элемента <include> целевое пространство имен включаемой схемы (атрибут targetNamespace) должно быть таким же, что и целевое пространство имен основной схемы, а при использовании элемента <import> можно ссылаться на компоненты схемы с произвольным целевым пространством имен.

Элемент <include> имеет следующие атрибуты:

- необязательный атрибут id — идентификатор элемента;
- обязательный атрибут schemaLocation — URI-адрес включаемой схемы.

Элемент <import> имеет следующие атрибуты:

- необязательный атрибут id — идентификатор элемента;
- необязательный атрибут namespace — целевое пространство имен компонентов схемы, на которые можно ссылаться в основной схеме;
- необязательный атрибут schemaLocation — URI-адрес схемы, на компоненты которой можно ссылаться в основной схеме.

С помощью элемента <redefine> можно переопределять компоненты внешней схемы, которая имеет такое же целевое пространство имен, что и основная схема.

Элемент <redefine> имеет атрибуты: необязательный id (идентификатор элемента) и обязательный schemaLocation (URI-адрес схемы, компоненты которой переопределяются).

Во все компоненты XML-схемы можно включать документацию с помощью элемента <annotation>, содержащего в свою очередь элементы <appinfo> и <documentation>.

Элемент <appinfo> XML-схемы позволяет давать информацию приложениям, использующим схему, а элемент <documentation> содержит текстовые комментарии. Информация элемента <appinfo> используется приложением в качестве инструкции по обработке. Элемент <appinfo> имеет необязательный атрибут source, указывающий URI-адрес приложения.

Элемент <documentation> имеет необязательные атрибуты source (URI-адрес приложения, использующего комментарии) и xml:lang (язык комментариев).

XML-схема определяет элементы и атрибуты для XML-документов, используя элементы <element> и <attribute>.

Элементы <element> и <attribute> называются глобальными, если они являются дочерними для элемента <schema>.

Элемент <element> (рис. 1.2) имеет следующие атрибуты.

- Необязательный атрибут abstract. Если true, тогда этот элемент является абстрактным и сам не может использоваться в XML-документе, а должен замещаться элементами, имеющими в качестве значения атрибута substitutionGroup — имя данного абстрактного элемента. По умолчанию значение атрибута abstract="false".
- Необязательный атрибут block ограничивает использование вместо данного элемента элементов с определенным типом наследования. Возможные значения:
  - #all — в XML-документах запрещена замена данного элемента его производными, созданными любым способом;

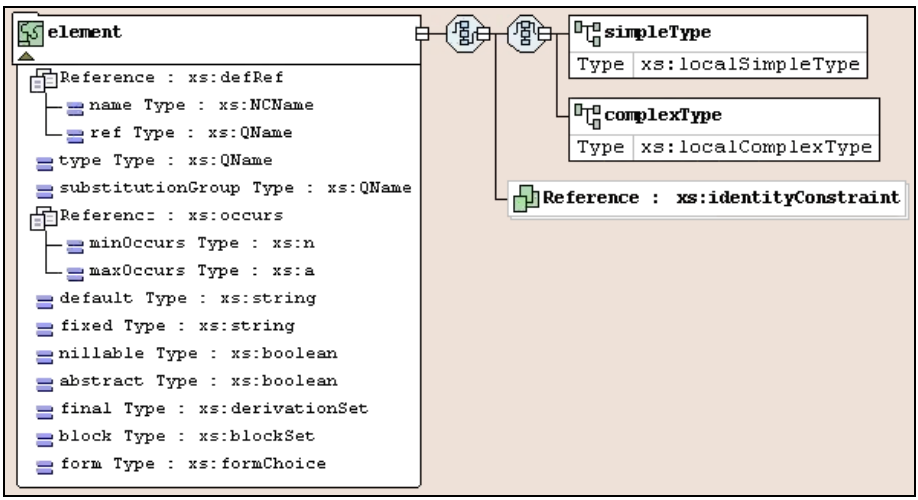


Рис. 1.2. Общая схема элемента `<element>`

- `extension` — запрещена замена в XML-документах данного элемента его производными элементами, полученными путем расширения типа данного элемента;
- `restriction` — запрещена замена в XML-документах данного элемента его производными элементами, полученными путем ограничения типа данного элемента;
- `substitution` — в XML-документах запрещена любая замена данного элемента.

❒ Необязательный атрибут `default` устанавливает для элементов с простым типом данных значение по умолчанию.

❒ Необязательный атрибут `final` ограничивает тип наследования данного элемента. Возможные значения:

- `#all` — для данного элемента в XML-схеме запрещено создание производных любым способом;
- `extension` — для данного элемента в XML-схеме запрещено создание производных путем расширения типа данного элемента;
- `restriction` — для данного элемента в XML-схеме запрещено создание производных путем ограничения типа данного элемента.

❒ Необязательный атрибут `fixed` для элемента с простым типом данных задает неизменяемое значение. Атрибут является взаимоисключающим с атрибутом `default`.

❒ Необязательный атрибут `form` задает форму элемента. Возможные значения:

- `qualified` — элемент необходимо уточнять префиксом пространства имен;
- `unqualified` — элемент не обязательно уточнять префиксом пространства имен.