Лекция 5.
Стандарт SOAP – протокол взаимодействия сервисов

SOAP - протокол обмена структурированными сообщениями в распределённой вычислительной среде. Поддерживается консорциумом W3C (http://www.w3.org/TR/soap/).

Протокол SOAP создан в 1998 году командой разработчиков под руководством Дейва Винера (Dave Winer), работавшей в корпорации Microsoft и фирме Userland, но затем передан в консорциум W3C.

Последняя версия стандарта на сегодняшний день - SOAP 1.2. В версии 1.1 SOAP расшифровывался как Simple Object Access Protocol — простой протокол доступа к объектам. Это название отражало его первоначальное назначение — обращаться к методам удаленных объектов. Сейчас назначение SOAP изменилось, поэтому разные разработчики предлагали свои варианты расшифровки. Поэтому в версии 1.2 аббревиатуру решили никак не расшифровывать.

Протокол SOAP не различает вызов процедуры и ответ на него, а просто определяет формат послания (message) в виде документа XML. Послание может содержать вызов процедуры, ответ на него, запрос на выполнение каких-то других действий или просто текст. Спецификацию SOAP не интересует содержимое послания, она задает только его оформление.

SOAP основан на языке XML и расширяет некоторый протокол прикладного уровня — HTTP, FTP, SMTP и т.д. Как правило чаще всего используется HTTP. Вместо использования HTTP для запроса HTML-страницы, которая будет показана в браузере, SOAP отправляет посредством HTTP-запроса XML-сообщение и получает результат в HTTP-отклике. Для правильной обработки XML-сообщения процесс-«слушатель» HTTP (напр. Apache или Microsoft IIS) должен предоставить SOAP-процессор, или, другими словами, должен иметь возможность обрабатывать XML.

SOAP является самой главной частью технологии Web-сервисов. Он осуществляет перенос данных по сети из одного места в другое.

SOAP обеспечивает доставку данных веб-сервисов. Он позволяет отправителю и получателю XML-документов поддерживать общий протокол передачи данных, что обеспечивает эффективность сетевой связи.

SOAP – это базовая однонаправленная модель соединения, обеспечивающая согласованную передачу сообщения от отправителя к получателю, потенциально допускающая наличие посредников, которые могут обрабатывать часть сообщения или добавлять к нему дополнительные элементы. Спецификация SOAP содержит соглашения по преобразованию однонаправленного обмена сообщениями в соответствии с принципом «запрос/ответ», а также определяет как осуществлять передачу всего XML-документа.

Как видно из рис.1 SOAP предназначен для поддержания независимого абстрактного протокола связи, обеспечивающего коммуникацию двух и более приложений, сайтов, предприятий и т.п., реализованных на разных технологиях и аппаратных средств.

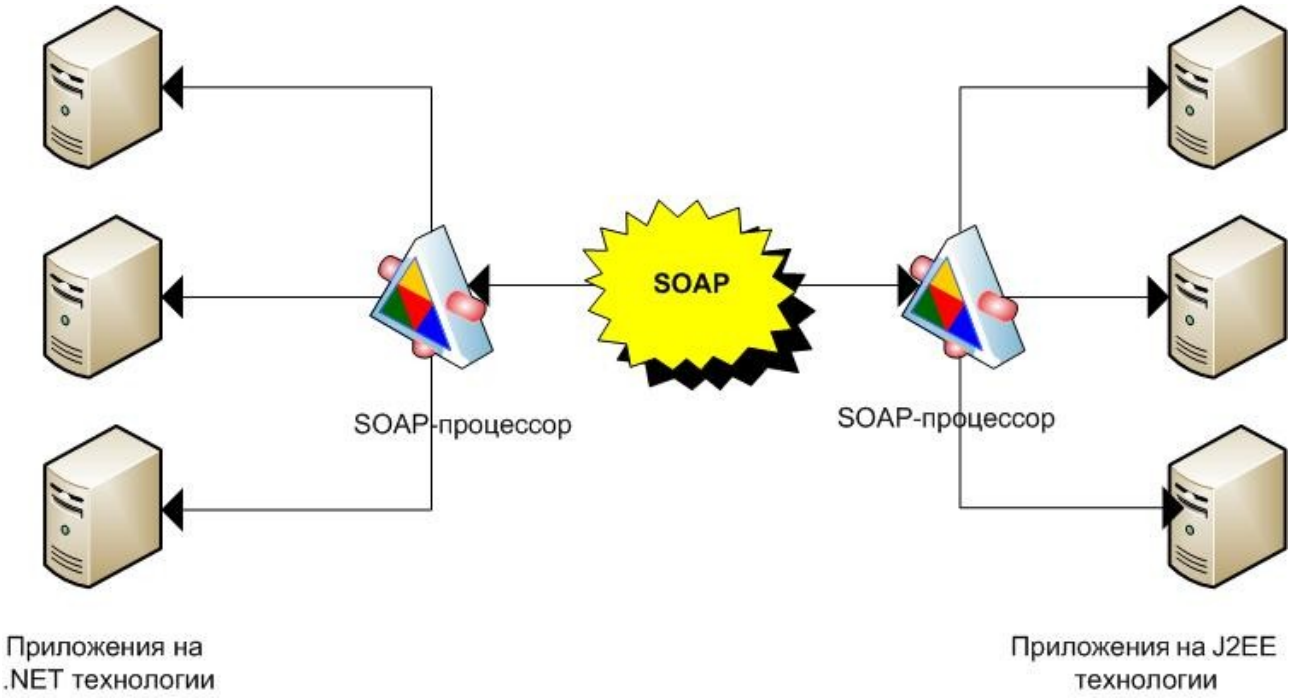

Рисунок 1.

5.1 Общая структура SOAP сообщения

SOAP-сообщение представляет собой XML-документ; сообщение состоит из трех основных элементов: конверт (SOAP Envelope), заголовок (SOAP Header) и тело (SOAP Body).

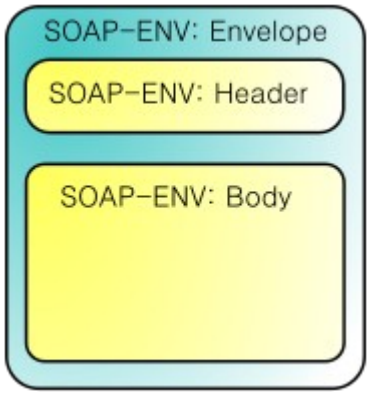

Рисунок 2. Структура SOAP сообщения

Пример SOAP сообщения:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
      xmlns:t="www.example.com">
      <SOAP-ENV:Header>
      </SOAP-ENV:Header>
      <SOAP-ENV:Body>
            <t:CurrentDate>
                  <Year>2011</Year>
                  <Month>February</Month>
                  <Day>12</Day>
                  <Time>18:02:00</Time>
            </t:CurrentDate>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Конверт (SOAP Envelope)

Является самым «верхним» элементом SOAP сообщения. Содержит корневой элемент XML-документа. Описывается с помощью элемента `Envelope` с обязательным пространством имен http://www.w3.org/2003/05/soap-envelope для версии 1.2 и http://schemas.xmlsoap.org/soap/ для версии 1.1.

У элемента `Envelope` могут быть атрибуты `xmlns`, определяющие пространства имен, и другие атрибуты, снабженные префиксами.

`Envelope` может иметь необязательный дочерний элемент `Header` с тем же пространством имен — заголовок. Если этот элемент присутствует, то он должен быть

первым прямым дочерним элементом конверта.

Следующий дочерний элемент конверта должен иметь имя `Body` и то же самое пространство имен - тело. Это обязательный элемент и он должен быть вторым прямым дочерним элементом конверта, если есть заголовок, или первым — если заголовка нет.

Версия 1.1 позволяла после тела сообщения записывать произвольные элементы, снабженные префиксами. Версия 1.2 это запрещает.

Элементы `Header` и `Body` могут содержать элементы из различных пространств имен.

Конверт изменяется от версии к версии. SOAP-процессоры, совместимые с версией 1.1, при получении сообщения, содержащего конверт с пространством имен версии 1.2, будут генерировать сообщение об ошибке. Аналогично для SOAP-процессоров, совместимых с версией 1.2. Ошибка — VersionMismatch.

Заголовок SOAP (SOAP Header)

Первый прямой дочерний элемент конверта. Не обязательный. Заголовок кроме атрибутов `xmlns` может содержать 0 или более стандартных атрибутов:

- `encodingStyle`
- `actor` (или `role` для версии 1.2)
- `mustUnderstand`
- `relay`

Атрибут `encodingStyle`

В SOAP-сообщениях могут передаваться данные различных типов (числа, даты, массивы, строки и т.п.). Определение этих типов данных выполняется в схемах XML (обычно — XSD). Типы, определенные в схеме, заносятся в пространство имен, идентификатор которого служит значением атрибута `encodingStyle`. Атрибут `encodingStyle` может появиться в любом элементе SOAP-сообщения, но версия SOAP 1.2 запрещает его появление в корневом элементе `Envelope`. Указанное атрибутом `encodingStyle` пространство имен

будет известно в том элементе, в котором записан атрибут, и во всех вложенных в него элементах. Какие-то из вложенных элементов могут изменить пространство имен своим атрибутом `encodingStyle`.

Стандартное пространство имен, в котором расположены имена типов данных SOAP 1.1, называется `http://schemas.xmlsoap.org/soap/encoding/`. У версии 1.2 — `http://www.w3.org/2003/05/soap-encoding`. Идентификатор того или иного пространства имен, в котором определены типы данных, обычно получает префикс enc или SOAP-ENC.

Атрибут `actor`

Тип данных URI. Задает адрес конкретного SOAP-сервера, которому предназначено сообщение.

SOAP-сообщение может пройти через несколько SOAP-серверов или через несколько приложений на одном сервере. Эти приложения выполняют предварительную обработку блоков заголовка послания и передают его друг другу. Все эти серверы и/или приложения называются SOAP-узлами (SOAP nodes). Спецификация SOAP не определяет правила прохождения послания по цепочке серверов. Для этого разрабатываются другие протоколы, например, Microsoft WS-Routing.

Атрибут actor задает целевой SOAP-узел — тот, который расположен в конце цепочки и будет обрабатывать заголовок полностью. Значение `http://schemas.xmlsoap.org/soap/actor/next` атрибута actor показывает, что обрабатывать заголовок будет первый же сервер, получивший его. Атрибут actor может встречаться в отдельных блоках заголовка, указывая узел-обработчик этого блока. После обработки блок удаляется из SOAP-сообщения.

В версии 1.2 атрибут actor заменен атрибутом `role`, потому что в этой версии SOAP каждый узел играет одну или несколько ролей. Спецификация пока определяет три роли SOAP-узла:

• Роль `http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver` играет конечный, целевой узел, который будет обрабатывать заголовок.

• Роль `http://www.w3.org/2003/05/soap-envelope/role/next` играет промежуточный или целевой узел. Такой узел может играть и другие, дополнительные роли.

• Роль `http://www.w3.org/2003/05/soap-envelope/role/none` не должен играть ни один SOAP-узел.

Распределенные приложения, исходя из своих нужд, могут добавить к этим ролям другие роли, например, ввести промежуточный сервер, проверяющий цифровую подпись и определить для него эту роль какой-нибудь строкой URI.

Значением атрибута role может быть любая строка URI, показывающая роль узла, которому предназначен данный блок заголовка. Значением по умолчанию для этого атрибута служит пустое значение, то есть, просто пара кавычек, или строка URI `http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver`.

Значение атрибута role показывает, что блок должен быть обработан узлом, играющим роль, определенную такой же строкой.

Атрибут `mustUnderstand`

Тип данных — boolean. По умолчанию 0. Если значение равно 1, то SOAP-узел при обработке элемента обязательно должен учитывать его синтаксис, определенный в схеме документа, или совсем не обрабатывать сообщение. Это повышает точность обработки сообщения.

В версии SOAP 1.2 вместо цифр нужно писать true или false.

Атрибут `relay`

Тип данных — boolean. Показывает, что заголовочный блок, адресованный SOAP-посреднику, должен быть передан дальше, если он *не* был обработан. Необходимо отметить, что если заголовочный блок обработан, правила обработки SOAP требуют, чтобы он был удален из уходящего сообщения. По умолчанию, необработанный заголовочный блок, предназначенный роли, которую исполняет SOAP-посредником, должен быть удален перед отправкой сообщения.

Все прямые дочерние элементы заголовка называются блоками заголовка (в версии 1.1. - статьями). Блоки заголовка используются для расширения сообщений децентрализованным способом путем добавления таких функций как аутентификация,

администрирование транзакций и т. п. Их имена обязательно должны помечаться префиками. В блоках заголовка могут быть атрибуты `role`, `actor` и `mustUnderstand`. Действие этих атрибутов относится только к данному блоку. Это позволяет обрабатывать отдельные блоки заголовка промежуточными SOAP-узлами, чья роль совпадает с ролью, указанной атрибутом `role`. Ниже дан пример такого блока:

```
<env:Header>
      <t:Transaction xmlns:t="http://example.com/transaction"
            env:role="http://www.w3.org/2003/05/soap-
envelope/role/ultimateReceiver"
            env:mustUnderstand="true">
            5
      </t:Transaction>
</env:Header>
```

Элементы, вложенные в блоки заголовка, уже не называются блоками. Они не могут содержать атрибуты `role`, `actor` и `mustUnderstand`.

Тело SOAP (SOAP Body)

Элемент `Body` обязательно записывается сразу за элементом `Header`, если он есть в сообщении, или первым в SOAP-сообщении, если заголовок отсутствует. В элемент `Body` можно вложить произвольные элементы, спецификация никак не определяет их структуру. Определен только один стандартный элемент, который может быть в теле сообщения - `Fault`, содержащий сообщение об ошибке.

5.2 **Обработка ошибок в SOAP-сообщениях**

Если SOAP-сервер, обрабатывая поступившее SOAP-сообщение, обнаружит ошибку, то он прекратит обработку и отправит клиенту SOAP-сообщение, содержащее один элемент `Fault` с сообщением об ошибке.

В версии 1.1 элемент `Fault` имел 4 дочерних элемента:

- код ошибки `faultcode` — предназначено для программы, обрабатывающей ошибки;
- описание ошибки `faultstring` – словесное описание типа ошибки, предназначено

для человека;

- место обнаружения ошибки `faultactor` – адрес URI сервера, заметившего ошибку. Промежуточные SOAP-узлы обязательно записывают этот элемент, целевой SOAP-сервер не обязан это делать;
- Детали ошибки `detail` — описывают ошибки, встреченные в теле `Body` послания, но не в его заголовке. Если при обработке тела ошибки не обнаружены, то этот элемент отсутствует.

Пример сообщения об ошибке:

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
      <env:Body>
            <env:Fault>
                  <faultcode>env:MustUnderstand</faultcode>
                  <faultstring>SOAP Must Understand Error</faultstring>
            </env:Fault>
      </env:Body>
</env:Envelope>
```

В версии SOAP 1.2 содержание элемента `Fault` изменилось . Как описано в пространстве имен `http://www.w3.org/2003/05/soap-envelope`, в него входят два обязательных элемента и три необязательных элемента.

Обязательные элементы:

- Код ошибки `code`. Он содержит обязательный вложенный элемент `value` с кодом ошибки и необязательный вложенный элемент `subcode`, также содержащий элемент `value` с уточняющим кодом ошибки и элемент `subcode`, и далее все повторяется рекурсивно.
- Причина ошибки `Reason`. Содержит необязательный атрибут `xml:lang`, указывающий язык сообщения, и произвольное число вложенных элементов с описанием ошибки.

Необязательные элементы:

- `Node` — адрес URI промежуточного SOAP-узла, заметившего ошибку.

- `Role` — роль SOAP-узла, заметившего ошибку.
- `Detail` — описание ошибки, замеченной при обработке тела `Body` послания, но не его заголовка.

В следующем примере показана ошибка, возникающая при попытке выполнения процедуры — имена аргументов процедуры неправильно записаны.

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
      xmlns:rpc='http://www.w3.org/2002/06/soap-rpc1>
      <env:Body>
            <env:Fault>
                  <env:Code>
                        <env:Value>env:Sender</env:Value>
                        <env:Subcode>
                              <env:Value>rpc:BadArguments</env:Value>
                        </env:Subcode>
                  </env:Code>
                  <env:Reason>Processing Error</env:Reason>
                  <env:Detail>
                        <e:myfaultdetails
xmlns:e="http://www.example.org/faults">
                              <message>Name does not match</message>
                              <errorcode>999</errorcode>
                        </e:myfaultdetails>
                  </env:Detail>
            </env:Fault>
      </env:Body>
</env:Envelope>
```

5.2.1. Типы ошибок

Список кодов ошибок постоянно меняется и расширяется. В версии 1.1 определены следующие типы ошибок:

- `VersionMismatch` — неправильное пространство имен (не та версия или его имя написано неправильно);
- `MustUnderstand` — блок заголовка, помеченный атрибутом `mustUnderstand` со

значением 1, не отвечает своему синтаксису, определенному в схеме документа;

- `Client` — документ XML, содержащий послание, неправильно сформирован и по этой причине сервер не может его обработать. Клиенту следует изменить послание.
- `Server` — сервер не может обработать правильно записанное послание по своим внутренним причинам.

Типы ошибок в версии 1.2:

- `VersionMismatch` — неправильное пространство имен (не та версия или его имя написано неправильно или или в сообщении встретилось имя элемента XML, не определенное в этом пространстве имен). В заголовок ответа сервер записывает элемент `<upgrade>`, перечисляющий вложенными элементами `<envelope>` правильные названия пространств имен, понимаемые сервером;
- `MustUnderstand` — блок заголовка, помеченный атрибутом `mustUnderstand` со значением true, не отвечает своему синтаксису, определенному в схеме документа. Сервер записывает в заголовок ответа элементы `<Misunderstood>`, атрибут `qname` которых содержит имя неправильного блока;
- `DataEncodingUnknown` — в послании встретились непонятные данные, может быть, они записаны в неизвестной кодировке;
- `Sender` — документ XML, содержащий послание, неправильно сформирован и по этой причине сервер не может его обработать. Клиенту следует изменить послание.
- `Receiver` — сервер не может обработать правильно записанное послание по своим внутренним причинам, например, отсутствует нужный XML-парсер.

Сервер может добавить к этим типам ошибок какие-то свои типы. Обычно они детализируют стандартные типы, и сообщения о них появляются в элементах `<subcode>`, как было показано выше.

Пример Fault сообщения с ошибкой `VersionMismatch`:

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
      <env:Header>
            <env:Upgrade>
```

```
                  <envelope qname="nsl:Envelope "
                        xmlns:nsl="http://www.w3.org/2003/05/soap-envelope"/>
                  <envelope qname="ns2:Envelope"
                        xmlns:ns2="http://schemas.xmlsoap.org/soap/envelope/"/>
            </env:Upgrade>
      </env:Header>
      <env:Body>
            <env:Fault>
                  <env:Code>
                        <env:Value>env:VersionMismatch</env:Value>
                  </env:Code>
                  <env:Reason>Version Mismatch</env:Reason>
            </env:Fault>
      </env:Body>
</env:Envelope>
```

Пример Fault сообщения с ошибкой `MustUnderstand`:

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env='http://www.w3.org/2003/05/soap-envelope'
      xmlns:flt='http://www.w3.org/2002/06/soap-faults'>
      <env:Header>
            <t:Misunderstood qname='t:Transaction'
                  xmlns:t='http://some.com/transaction'/>
      </env:Header>
      <env:Body>
            <env:Fault>
                  <env:Code>
                        <env:Value>env:MustUnderstand</env:Value>
                  </env:Code>
                  <env:Reason>
                        One or more mandatory headers not understood
                  </env:Reason>
            </env:Fault>
      </env:Body>
</env:Envelope>
```

5.3 SOAP-сообщения с вложениями

Существуют ситуации, когда клиент и сервер должны обмениваться данными в формате, отличном от текстового. Это могут быть данные в форматах каких-то приложений, мультимедийные данные и т.п. С точки зрения обмена данными все нетекстовые данные рассматриваются как данные в двоичных кодах.

Двоичные данные включаются в сообщение в виде «вложения». В 2000 году консорциумом W3C выпущена спецификация «SOAP-сообщения с вложениями» (SOAP with Attachments) (http://www.w3.org/TR/SOAP-attachments/), опирающаяся на версию SOAP 1.1. В спецификации описаны правила включения SOAP-сообщения в MIME-сообщение типа multipart/related и правила пересылки его по протоколу HTTP.

Структура SOAP-сообщения с вложениями показана на рис. 3.

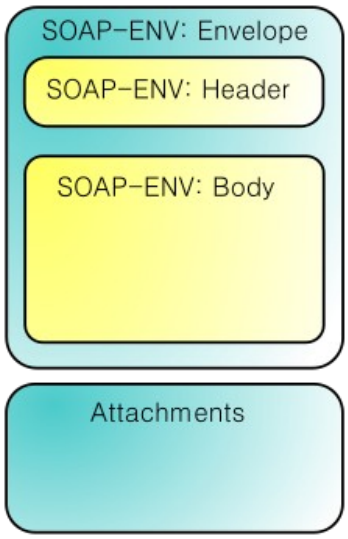

Рисунок 3. Структура SOAP-сообщения с вложениями

Этот протокол определяет пересылку SOAP-сообщения внутри MIME-сообщения, состоящего из нескольких частей. Первая часть MIME-сообщения сообщения - часть SOAP - содержит XML: конверт SOAP с вложенными в него заголовком и телом сообщения. Остальные части - вложения - содержат данные в любом формате, двоичном или текстовом. Каждая часть предваряется MIME-заголовком, описывающим формат данных части и содержащим идентификатор части (`Content-ID`). По этому идентификатору тело SOAP-сообщения может ссылаться на вложения (`href`). Приведенный ниже фрагмент должен дать представление о структуре SOAP-сообщения с вложениями. Жирным шрифтом в нем

выделен идентификатор вложения и ссылка на него из основной части сообщения.

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary;
type="application/xop+xml";
start="<soapmsg.xml@leonardo.com>";
start-info="text/xml"
Content-Description: An XML document with binary data in it
--MIME_boundary

Content-Type: application/xop+xml;
charset=UTF-8;
type="text/xml"
Content-Transfer-Encoding: 8bit
Content-ID: <soapmsg.xml@daily-moon.com>
<env:Envelope . . . >
  <env:Header>
    . . .
  </env:Header>
  <env:Body>
    . . .
    <xop:include
       xmlns:xop='http://www.w3.org/2004/08/xop/include'
       href='cid:http://leonardo.com/mona_liza.jpg'/>
    . . .
  </env:Body>
</env:Envelope>
--MIME_boundary

Content-Type: image/jpg
Content-Transfer-Encoding: binary
Content-ID: <http://leonardo.com/mona_liza.jpg>
// двоичное представление файла изображения
--MIME_boundary--
```

5.4 Средства технологии Java для работы с SOAP

Для работы с SOAP в Java предусмотрены следующие средства:

- Java API for XML-based RPC (JAX-RPC),
- SOAP with Attachments API for Java (SAAJ),
- Java API for XML Messaging (JAXM).

JAX-RPC. В основе взаимодействия Web-сервисов и клиентов лежит JAX-RPC (JSR-101) - технология, которая использует вызовы удаленных процедур (RPC) и XML. Данная технология на сегодняшний день является уже устаревшей и исключена из J2EE версии 6.

Хотя JAX-RPC предназначен прежде всего для RPC-ориентированного стиля обмена, он применим также и для документо-ориентированного стиля. Часто используемый в

распределенной клиент-серверной модели, механизм RPC позволяет клиентам выполнять процедуры на другой системе. JAX-RPC адаптирует имеющуюся в Java технологию вызова удаленных методов (RMI - Remote Method Invocation) для обмена по XML-протоколам. Архитектура JAX-RPC показана на рис.4:

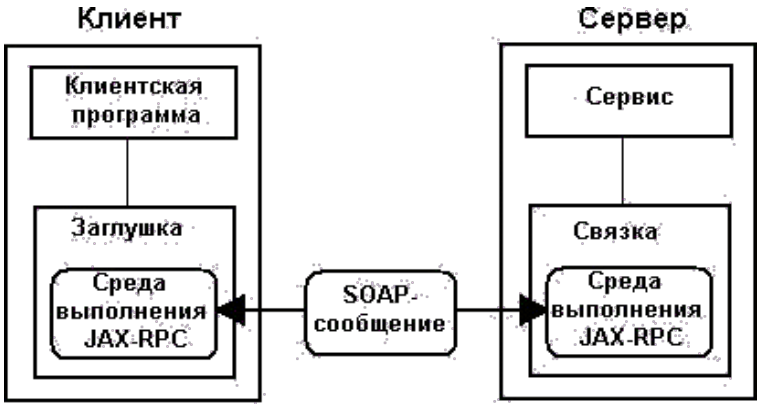

Рисунок 4. Архитектура JAX-RPC

Основой взаимодействия клиента и сервиса в JAX-RPC является *интерфейс конечной точки* (удаленный интерфейс в терминах Java RMI), который представляет методы сервиса, доступные для клиента. Клиент обращается к заглушке - локальному объекту, который реализует интерфейс конечной точки и является как бы представителем сервиса на машине клиента. Заглушка обращается к серверу через *среду выполнения* JAX-RPC. Среда выполнения преобразует вызов в SOAP-сообщение, при этом JAX-RPC отображает типы языка Java на определения XML/WSDL - так же, как это происходит при преобразовании Java в WSDL. Отклик сервиса среда также преобразует в данные для Java-программы. Здесь же применяются и специальные классы - сериализаторы и десериализаторы, - преобразующие в XML данные, представленные нетривиальными классами Java и компонентами JavaBeans. На стороне сервера также присутствует связующий компонент - связка, который взаимодействует со средой выполнения JAX-RPC на стороне сервера и выполняет аналогичную работу для сервиса. Хотя как клиентская, так и серверная часть взаимодействия достаточно сложны (в их состав помимо основных классов сервиса и клиента входят интерфейс конечной точки, заглушки и связки, сериализаторы и десериализаторы), значительная часть кодов генерируется автоматически средствами разработки (утилитой wscompile). В клиенте Web-сервиса кодируются только следующие операции:

- создается объект типа `Stub`, представляющий сервис на стороне клиента;

- методом `_setProperties` на объекте типа `Stub` устанавливается адрес конечной точки;
- объект типа `Stub` приводится к типу интерфейса конечной точки;
- на приведенном объекте выполняются методы сервиса.

Клиенты, создаваемые средствами JAX-RPC, могут быть статическими или динамическими. В статическом клиенте заглушка генерируется при компиляции клиентской программы. В динамическом клиенте WSDL-описание заранее неизвестно. Динамические клиенты обращаются к Web-сервису по его WSDL-описанию. При этом на сервере создается заглушка и пересылается на клиентскую машину. Методы, позволяющие клиенту использовать для обращения к сервису WSDL-описание, предоставляются интерфейсом Service.

Клиент может не иметь ни заранее созданных заглушек, ни WSDL-описания сервиса. Для такого случая JAX-RPC предоставляет интерфейс Call, позволяющий вызывать методы Web-сервиса, зная только его адрес, название и сигнатуру метода.

JAX-RPC полностью избавляет программиста от необходимости формирования самого сообщения в формате SOAP. При необходимости, однако, программист может сам формировать SOAP-сообщения, используя средства пакета SAAJ. (Этот же пакет использует «за кулисами» и JAX-RPC.)

SAAJ. Открытая спецификация фирмы Sun Microsystems SAAJ (http://java.sun.com/xml/downloads/saaj.html) использует модель документа как дерева объектов DOM, поэтому интерфейсы пакета SAAJ расширяют интерфейсы DOM, определенные в базовом пакете `org.w3c.dom`: интерфейс `Node` SAAJ расширяет интерфейс DOM `Node`; интерфейс `SOAPElement` SAAJ расширяет интерфейсы DOM `Node` и `Element`; класс `SOAPPart` SAAJ реализует интерфейс DOM `Document`; интерфейс `Text` SAAJ расширяет интерфейс DOM `Text`. В пакете имеется также ряд интерфейсов, расширяющих интерфейс `SOAPElement` и соответствующих элементам SOAP-сообщения: `SOAPEnvelope`, `SOAPBody`, `SOAPHeader` и другие. Методы этих интерфейсов дают возможность извлекать из элементов SOAP-сообщения их содержимое или добавлять содержимое в элемент. Начальным объектом при формировании сообщения является объект типа SOAPMessage, который создается фабричным методом. Из сообщения затем можно

получить ссылку на часть сообщения (`SOAPPart`), из части - ссылку на конверт, из конверта - ссылки на заголовок и тело и т.д. Поскольку узлы и элементы SAAJ реализуют интерфейсы DOM `Node` и `Element`, работать с содержимым сообщения SOAP можно, используя только API DOM или только API SAAJ, или переключаясь с одного API на другой.

Все сообщения в SAAJ посылаются и принимаются через соединения. Соединения представляются объектом `SOAPConnection`, получаемого фабричным методом. Сообщения посылаются через объект `SOAPConnection` при помощи метода `call`, аргументами которого являются сообщение и адрес. Метод посылает сообщение и возвращает ответ, между посылкой сообщения и получением ответа метод блокируется.

SAAJ обеспечивает соединение клиента и сервиса «точка в точку» и обмен сообщениями «запрос-ответ». До получения ответа клиент блокируется. Однако сервис не всегда может быть доступен, и большое число прикладных задач и не требует ответа от сервиса. Для того, чтобы допускать продолжение работы клиента при отсутствии ответа от сервиса необходим механизм асинхронного обмена сообщениями. Модель обмена «запрос-ответ» также может быть реализована в асинхронном режиме. Эту задачу решает пакет JAXM, который обычно используется в приложениях совместно с SAAJ. Подготовка сообщения производится средствами SAAJ. Посылается же сообщение через посредство провайдера сообщений - некоторого кода, который обеспечивает доставку сообщения адресату. Соединение с провайдером - объект типа `ProviderConnection` - получается фабричным методом или через службу именования (JNDI - Java Naming and Directory Interface). Сообщение отсылается адресату методом провайдера send.

Получатель сообщения должен реализовывать интерфейс `OnewayListener` (если он не отправляет ответа) или интерфейс `ReqRespListener` (если он отправляет ответное асинхронное сообщение). Оба интерфейса описывают обязательный для реализации метод `onMessage`, который вызывается при получении сообщения. Аргументом метода является объект типа `SOAPMessage` - полученное сообщение. В первом интерфейсе метод не возвращает ничего, а во втором он возвращает также объект типа `SOAPMessage` - ответное сообщение.

Литература

1. Стандарт SOAP - http://www.w3.org/TR/soap/
2. SOAP Версия 1.2 Часть 0: Учебник для начинающих : http://www.w3.org/2002/07/soap-translation/russian/part0.html
3. Хабибуллин И.Ш. Разработка Web-служб средствами Java. — СПб. : БХВ-Петербург, 2003. — 400 с: ил.
4. Технологии и средства консолидации информации: Учебное пособие. Деревянко А.С., Солощук М.Н. - Харьков: НТУ "ХПИ", 2008. - 432с.