

**SOAP**

# SOAP

SOAP — протокол обмена структурированными сообщениями в распределённой вычислительной среде. Первоначально SOAP предназначался, в основном, для реализации удалённого вызова процедур (RPC), а название было аббревиатурой: Simple Object Access Protocol — простой протокол доступа к объектам. Сейчас протокол используется для обмена произвольными сообщениями в формате XML, а не только для вызова процедур.

# SOAP

SOAP может использоваться с любым протоколом прикладного уровня: SMTP, FTP, HTTP и др. Однако его взаимодействие с каждым из этих протоколов имеет свои особенности, которые должны быть определены отдельно. Чаще всего SOAP используется поверх HTTP.

SOAP является одним из стандартов, на которых базируется технология веб-сервисов.

# Структура протокола

**Сообщение SOAP выглядит так:**

SOAP- конверт

SOAP-заголовок

Элемент заголовка 1

Элемент заголовка 2

...

Элемент заголовка N

Тело SOAP

Элемент тела N

...

Элемент тела 2

Элемент тела 1

# Пример

**Пример SOAP-запроса на сервер интернет-магазина:**

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetails
      xmlns="http://warehouse.example.com/ws">
      <productID>12345</productID>
    </getProductDetails>
  </soap:Body>
</soap:Envelope>
```

# Пример

## Ответ:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
      <getProductDetailsResult>
        <productID>12345</productID>
        <productName>Стакан граненый</productName>
        <description>Стакан граненый. 200 мл.</description>
        <price>9.95</price>
        <inStock>true</inStock>
      </getProductDetailsResult>
    </getProductDetailsResponse>
  </soap:Body>
</soap:Envelope>
```

# Недостатки

Использование SOAP для передачи сообщений увеличивает их объём и снижает скорость обработки. В системах, где скорость важна, чаще используется пересылка XML документов через HTTP напрямую, где параметры запроса передаются как обычные HTTP параметры.

Хотя SOAP является стандартом, различные программы часто генерируют сообщения в несовместимом формате. Например, запрос сгенерированный AXIS-клиентом, не будет понят сервером WebLogic.

# Заголовки SOAP-сообщения

Заголовочный элемент SOAP не является обязательным, однако он был включен в пример 1 для того, чтобы объяснить некоторые функции SOAP. Заголовки SOAP являются расширением, предоставляющим способ передачи в SOAP-сообщениях информации, вообще говоря не являющейся полезной для приложения. Подобная "контрольная" информация включает, например, директивы прохождения сообщения или контекстную информацию, относящуюся к обработке сообщения. Это позволяет подстраивать SOAP-сообщения под каждое конкретное приложение.



# Заголовки SOAP-сообщения

Следующие непосредственно за `env:Header` дочерние элементы называются заголовочными блоками. Они представляют логическую группировку данных, которые, как показано позже, могут быть индивидуально адресованы SOAP-узлам, встречаемым сообщением на пути от отправителя к конечному получателю.

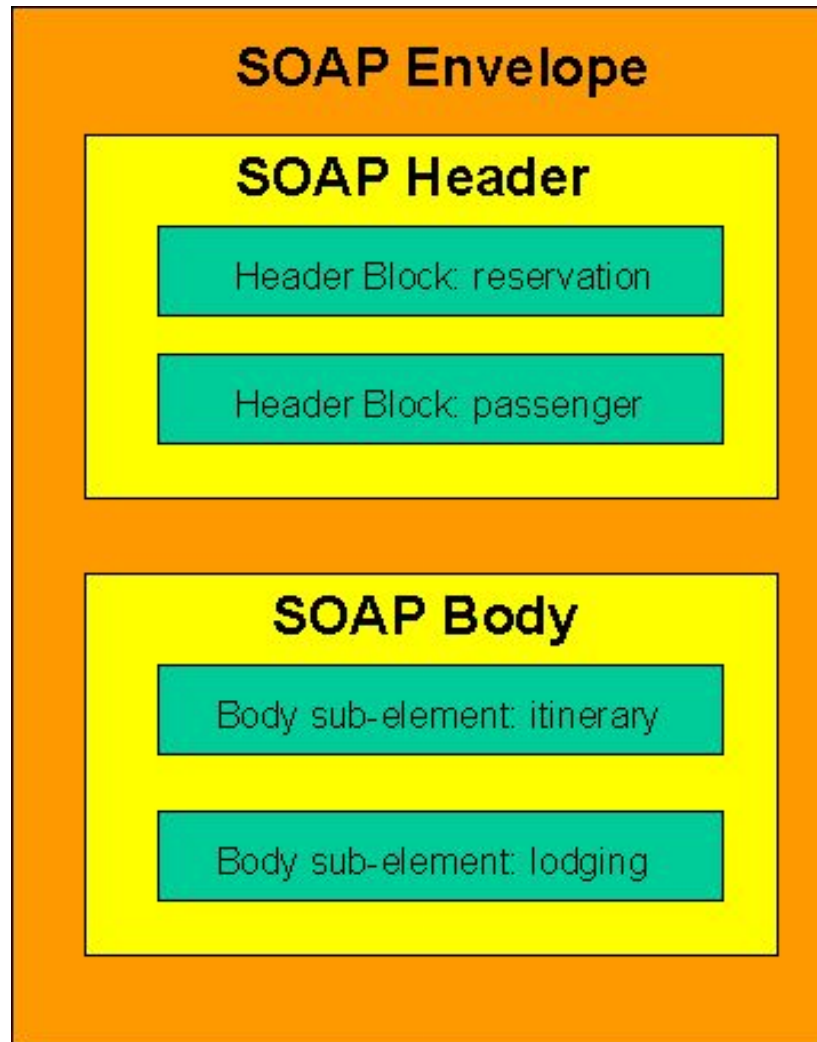
# Заголовки SOAP-сообщения

Заголовки SOAP были созданы в предположении появления различных вариантов использования SOAP, многие из которых будут вовлекать во взаимодействие другие обрабатывающие SOAP-сообщения узлы, называемые SOAP-посредниками - на пути сообщения от начального отправителя SOAP-сообщения до конечного SOAP-получателя. Это позволяет SOAP-посредникам предоставлять дополнительные сервисы. Заголовки, как показано далее, могут быть просмотрены, вставлены, удалены или пересланы SOAP-узлами, встреченными на пути SOAP-сообщения. (Однако, необходимо помнить, что спецификации SOAP не описывают содержимое заголовочных элементов, или то, как SOAP-сообщения маршрутизируются между узлами. Они также не описывают каким образом определяется маршрут сообщения и т. д. Эти вопросы решаются приложением в целом и могут быть предметом рассмотрения других спецификаций.)

# Тело SOAP-сообщения

Тело SOAP-сообщения является обязательным элементом внутри `env:Envelope`, содержащим основную информацию SOAP-сообщения, которая должна быть передана из начальной точки пути сообщения в конечную.

# Тело SOAP-сообщения



# Тело SOAP-сообщения

В примере 1, заголовок содержит два заголовочных блока, каждый из которых определен в собственном пространстве имен XML, и отражает некоторый аспект общей обработки тела SOAP-сообщения. Для приложения бронирования путешествия, подобная, принадлежащая к запросу в целом, "метаянформация" содержится в заголовочном блоке **reservation**, который представляет собой ссылку на этот экземпляр запроса, а также содержит его временную отметку. "Метаянформация", служащая для идентификации будущего путешественника, содержится в блоке **passenger**.

# Тело SOAP-сообщения

Заголовочные блоки **reservation** и **passenger** должны обрабатываться следующим SOAP-посредником, встреченным на пути сообщения, либо, в случае отсутствия узла-посредника, конечным получателем сообщения. На тот факт, что сообщение адресовано следующему SOAP-узлу, встреченному на пути сообщения, указывает присутствие атрибута **env:role** со значением "<http://www.w3.org/2003/05/soap-envelope/role/next>" (здесь и далее просто "next"). Этот атрибут реализует роль, исполнять которую обязаны все SOAP-узлы. Присутствие же атрибута **env:mustUnderstand** со значением "true" указывает на то, что узел (узлы), обрабатывающий заголовочные блоки, должен обрабатывать их в строгом соответствии с их спецификациями либо не обрабатывать SOAP-сообщение вовсе и выдать сообщение об ошибке.

# Тело SOAP-сообщения

Элемент `env:Body` и ассоциированные с ним дочерние элементы `itinerary` и `lodging`, вовлечены в обмен информацией между начальным отправителем SOAP-сообщения и SOAP-узлом на его пути, выступающим в роли конечного получателя SOAP-сообщения, которым является сервис продажи билетов. Поэтому элемент `env:Body` с его содержимым всецело адресован конечному получателю и должен быть им понят. Средства, посредством которых SOAP-узел может выполнять эту роль, не определяются спецификацией SOAP. Они определяются общей семантикой приложения и ассоциированного с ним потоком сообщений.

# Тело SOAP-сообщения

SOAP-сообщение, подобное представленному в примере, может быть передано посредством различных нижележащих протоколов и использоваться в различных шаблонах обмена сообщениями. Например, для web-доступа к сервису продажи билетов, оно может быть помещено в тело HTTP-запроса POST. В случае привязки к другому протоколу, оно может быть отправлено в email-сообщении. SOAP-сообщения могут передаваться посредством различных нижележащих протоколов. Сейчас же предположим, что механизм для передачи сообщения уже существует, и сконцентрируемся на деталях SOAP-сообщений и их обработки.



# Диалоговый обмен SOAP-сообщениями

Пример 2 демонстрирует SOAP-сообщение, полученное от сервиса продажи билетов в ответ на запрос о бронировании, реализованный сообщением примера 1.

# Диалоговый обмен SOAP-сообщениями

Обмен сообщениями в Примерах 1 и 2 является случаем, когда посредством SOAP-сообщений осуществляется обмен XML-контентом, удовлетворяющим некоторой определенной приложением схеме.

Однако легко видеть, как подобный обмен может быть построен с помощью двунаправленного "диалогового" шаблона обмена сообщениями. Пример 3 демонстрирует SOAP-сообщение, отправленное приложением бронирования путешествия в ответ на сообщение примера 2 и содержащее аэропорт, выбранный из списка доступных аэропортов. Заголовочный блок **reservation** с тем же значением субэлемента **reference** содержится в каждом сообщении этого диалога, тем самым предоставляя, в случае необходимости, возможность коррелировать сообщения на уровне приложения.

# Вызовы удаленных процедур

Приложение бронирования путешествия предоставляет информацию о кредитной карточке, а также информацию об успешном завершении различных операций, происходящих в результате снятия денег со счета с помощью кредитной карточки и возврата кода бронирования. Это взаимодействие между приложением бронирования путешествия и сервисом продажи билетов с целью получения брони и ее оплаты реализовано посредством SOAP RPC.

# Вызовы удаленных процедур

Для вызова SOAP RPC требуется следующая информация:

1. Адрес SOAP-узла места назначения;
2. Имя процедуры либо метода;
3. Наименования и значения всех аргументов, передаваемых процедуре или методу вместе с выходными параметрами и возвращаемым значением;
4. Четкое разделение аргументов, используемых для идентификации web-ресурса, являющегося действительным местом назначения RPC, от аргументов, содержащих данные и контрольную информацию, используемых для обработки вызова ресурсом места назначения RPC.
5. Определение шаблона обмена сообщениями, а также так называемого "Web-метода" (о нем будет рассказано несколько позже), которые будут использоваться для передачи RPC.
6. Данные, которые могут быть переданы как часть заголовочных блоков SOAP. Эти данные не являются обязательными.

# Сценарии обработки сообщений об ошибках

SOAP-элемент **env:Body** имеет также еще одну характерную роль - он может содержать информацию об ошибке. Элемент **env:Fault** содержит два обязательных субэлемента, **env:Code** и **env:Reason**, и (необязательно) специфичную для приложения информацию в субэlemente **env:Detail**. Другой необязательный субэлемент **env:Node** посредством URI определяет SOAP-узел, сгенерировавший ошибку. Отсутствие этого субэлемента означает, что ошибка была сгенерирована конечным получателем сообщения. Существует также еще один необязательный субэлемент, **env:Role**, определяющий роль, исполняемую сгенерировавшим ошибку узлом.

Пример 5.

# Модель обработки SOAP

Модель обработки SOAP описывает действия SOAP-узла при получении SOAP-сообщения.

Атрибут "role". Роль узла.

Атрибут "mustUnderstand"

# Модель обработки SOAP

Узел	посредник	конечный получатель
mustUnderstand		
"true"	должен обработать	должен обработать
"false"	может обработать	может обработать
отсутствует	может обработать	может обработать

# Модель обработки SOAP

Атрибут "relay". Передавать ли дальше необработанный заголовок.



# HTTP-привязка SOAP

HTTP имеет хорошо известную модель взаимодействия и шаблон обмена сообщениями. Клиент идентифицирует сервер по URI, подсоединяется к нему с помощью TCP/IP сети, отправляет HTTP-сообщение-запрос и получает HTTP-сообщение-отклик по тому же TCP-соединению. HTTP полностью коррелирует сообщение-запрос и соответствующий ему сообщение-отклик, поэтому приложение, использующее эту привязку, может реализовать корреляцию между отправленным в теле HTTP-запроса SOAP-сообщением и SOAP-сообщением, возвращенным в качестве HTTP-отклика. Подобным образом, HTTP идентифицирует конечный сервер по URI, URI-запросу, который может также служить идентификатором SOAP-узла сервера.

# HTTP-привязка SOAP

HTTP-привязка использует функцию SOAP Web-метода, позволяющую приложениям выбирать один из так называемых Web-методов - GET или POST - чтобы использовать для обмена сообщениями с помощью HTTP. Кроме того, она использует два шаблона обмена сообщениями, которые дают приложениям два способа обмена SOAP-сообщениями посредством HTTP: 1) использование HTTP-метода POST для передачи SOAP-сообщений в теле HTTP-запроса и HTTP-отклика, и 2) использование HTTP-метода GET в HTTP-запросе для возвращения SOAP-сообщения в теле HTTP-отклика. Первый шаблон использования является HTTP-реализацией функции привязки - шаблона обмена SOAP-сообщениями типа **"запрос-отклик"**, второй - шаблона обмена SOAP-сообщениями типа **"отклик"**.

# HTTP-привязка SOAP

Цель разработки этих двух способов - реализовать две парадигмы взаимодействия, одинаково хорошо подходящие для World Wide Web. Первый тип взаимодействия позволяет использовать данные в теле HTTP-метода POST для создания или изменения состояния ресурса, идентифицируемого по URI, в соответствии с которым направлен HTTP-запрос. Второй тип шаблона взаимодействия дает возможность использовать HTTP-запрос GET для получения представления о ресурсе без какого-либо изменения его состояния. В первом случае касающийся SOAP аспект вопроса состоит в том, что тело HTTP-запроса POST является SOAP-сообщением, которое кроме того, что должно быть обработано (согласно модели обработки SOAP) в соответствии со специфичной для приложения обработкой, должно также соответствовать семантике POST. Во втором случае типичной реализацией является получение представления запрашиваемого ресурса в виде SOAP-сообщения, а не в виде HTML- или XML-документа.

# Использование в SOAP HTTP-метода GET

GET /travelcompany.example.org/reservations?code=FT35ZBQ HTTP/1.1

Host: travelcompany.example.org

Accept: text/html;q=0.5, application/soap+xml

# Использование в SOAP HTTP-метода POST

Смотри пример 6.

# Использование SOAP поверх Email

Разработчики приложений могут использовать также email-инфраструктуру для передачи SOAP-сообщений, причем как текст email-сообщений так и их вложения. Примеры, приведенные ниже, иллюстрируют такой метод передачи SOAP-сообщений, однако они не должны трактоваться как некие стандартные способы реализации этого метода. Спецификации SOAP Версия 1.2 не специфицируют подобную привязку. Хотя существует неофициальный W3C Note, описывающий email-привязку для SOAP. Его основная цель - продемонстрировать применение общей Структуры Протокольной Привязки SOAP.

# Использование SOAP поверх Email

Разработчики приложений могут использовать также email-инфраструктуру для передачи SOAP-сообщений, причем как текст email-сообщений так и их вложения. Примеры, приведенные ниже, иллюстрируют такой метод передачи SOAP-сообщений, однако они не должны трактоваться как некие стандартные способы реализации этого метода. Спецификации SOAP Версия 1.2 не специфицируют подобную привязку. Хотя существует неофициальный W3C Note, описывающий email-привязку для SOAP. Его основная цель - продемонстрировать применение общей Структуры Протокольной Привязки SOAP.