

## Что такое WSRF, Часть 1: Использование WS-ResourceProperties

Бабу Сандарам

[ibm.com/developerWorks](http://ibm.com/developerWorks)

\* \* \*

## Understanding WSRF, Part 1: Using WS-ResourceProperties

Babu Sundaram

[babu@cs.uh.edu](mailto:babu@cs.uh.edu)

## Содержание

Содержание .....	2
1. Прежде всего .....	3
1.1. На кого рассчитан этот учебный материал? .....	3
1.2. О чем этот учебный материал? .....	3
1.3. Что должно быть под рукой .....	4
1.4. Об авторе .....	4
2. Общий обзор .....	4
2.1. О сущности грид-компьютинга .....	4
2.2. Использование Web-служб: надежды и трудности .....	5
2.3. Ресурсы с состоянием .....	6
2.4. Web-службы + ресурсы с состоянием = WS-ресурсы .....	6
2.5. Свойства WS-ресурса .....	7
2.6. Воспользуйтесь WSRF .....	7
3. Что нужно знать о WSDL .....	8
3.1. Что такое WSDL, и зачем оно мне? .....	8
3.2. Сообщения и типы .....	9
3.3. Типы портов и операции .....	11
3.4. Службы и связывание .....	11
4. Что вам необходимо знать о WS-адресации .....	13
4.1. Что такое WS-адресация, и зачем она мне? .....	13
4.2. Рассмотрим WS-адресацию .....	13
4.3. Ссылки на крайнюю точку .....	14
5. Создание WS-ресурса .....	14
5.1. Что такое WS-ресурс? .....	14
5.2. Документ свойств ресурса .....	15
5.3. Переопределение WS-ресурса .....	15
5.4. Объединение данной Web-службы с данным ресурсом: WSDL-файл .....	16
5.5. Добавление ресурса к WSDL-файлу .....	17
5.6. Запрос нового спутника .....	19
5.7. SOAP-запрос .....	21
5.8. SOAP-ответ .....	22
6. Как работать с WS-ресурсом: доступ к свойствам .....	22
6.1. Что мы пытаемся сделать .....	22
6.2. Выбор значения свойства .....	23
6.3. Полный SOAP-запрос .....	23
6.4. Получение свойства ресурса .....	24
6.5. WSDL-файл .....	25
6.6. Запрос множества свойств .....	28
6.7. Получение множества свойств .....	28
6.8. WSDL-файл .....	29
6.9. Использование для запросов выражений XPath .....	31
6.10. Результаты запроса .....	32
6.11. WSDL-файл .....	33
7. Как работать с WS-ресурсом: установка значений свойств .....	35
7.1. Что мы пытаемся сделать .....	35
7.2. Добавление свойства .....	35
7.3. Результат добавления свойства .....	36
7.4. Изменение значения свойства .....	36
7.5. Удаление свойства .....	37
7.6. WSDL-файл .....	38
8. Итоги .....	42
8.1. Учебные ресурсы .....	43
8.2. WSRF и относящиеся к нему спецификации .....	43
8.3. Web-службы и относящиеся к ним спецификации .....	43

## 1. Прежде всего

### 1.1. На кого рассчитан этот учебный материал?

В этой публикации вводятся концепции, лежащие в основе платформы WS-ресурс (WSRF). В частности, объясняется, что это такое – WS-ресурс, как его создать и как с ним работать.

Публикация предназначена пользователям, которые хотят создавать приложения, опирающиеся на Web-службы и предполагающие использование «ресурсов, обладающих состоянием» (stateful resources). В самом общем случае, приложениями этого типа являются грид-службы, но обсуждаемые здесь понятия применимы к любым WS-приложениям, использующим понятие «состояния». Такими WS-приложениями являются приложения, использующие файлы или базы данных. Файлы или базы данных существуют и тогда, когда к ним не обращаются. (Более подробное объяснение можно прочитать в разделе «Ресурсы с состоянием».)

Так как это концептуальное пособие, то умение программировать не предполагается, но весьма желательным считается общее знакомство с такими программными понятиями как объекты и свойства. Впрочем, достаточно строго обсуждается создание WSDL-файла. WSDL – язык описания Web-служб. Базовые понятия WSDL разъясняются, но общее знакомство с языком XML и Web-службами было бы весьма желательным. (Введение в XML и Web-службы содержится в разделе «Учебные курсы».)

### 1.2. О чем этот учебный материал?

Эта публикация является первой частью в серии из четырех частей, в которой обсуждаются базовые понятия инфраструктуры WS-ресурс (WSRF). WSRF это множество спецификаций, обеспечивающих стандартный способ обработки ресурсов «с состоянием» в среде WS-приложения, в принципе не имеющей «состояния». Этот так называемый ресурс с состоянием может быть чем угодно, от базы данных до электронного датчика. И вообще, вы можете использовать WSRF при работе с любой сущностью, манипулирование которой происходит путем изменения ее свойств.

WS-ресурс – это объединение ресурса с состоянием, такого, как база данных или некоторое информационное устройство, и Web-службы, с которой устройство взаимодействует. В нашем пособии объясняется процесс создания рассматриваемых WS-ресурсов и работы с ними (выяснению и модификации свойств WS-ресурса, определяющих его состояние). Мы рассмотрим следующие темы.

- Общий обзор WSRF-спецификаций
- Общий обзор прикладной WSRF-системы, на которой мы будем производить демонстрации в этой серии учебных пособий
- Основные сведения о WSDL
- Основные сведения о WS-адресации, используемой при поиске WS-ресурса
- Как создать WS-ресурс
- Как получить и модифицировать свойства WS-ресурса

Отметим, что WSRF-спецификации определяют структуру WSDL-файла, который описывает операции создания WS-ресурса и работу с ним. В дальнейшем, этот WSDL-файл может быть использован в приложении для любой языковой среды реализации. Мы опишем процесс создания WSDL-файла и покажем результирующие SOAP сообщения.

Еще раз обратим Ваше внимание на то, что WSRF-спецификации определяют то, *что* должно быть сделано, но не определяют, *как* это должно быть сделано. Фактическая реализация содержательной стороны целиком перекладывается на само приложение. В части 4 данной серии мы рассмотрим реализацию на базе классов Core Java WSRF (Globus Alliance).

В качестве примера рассмотрим группу искусственных спутников, вращающихся вокруг Земли и производящих различные замеры. Мы создадим WS-ресурс, относящийся к одному из этих спутников. Затем мы запросим значения свойств созданного WS-ресурса и просмотрим результаты измерений. Мы переместим спутник в небе и изменим его ориентацию, для этого изменим некоторые свойства WS-ресурса.

### **1.3. Что должно быть под рукой**

Так как это пособие не предполагает программирования, единственное, что требуется, это текстовый редактор для создания WSDL-файлов. Впрочем, мы будем ссылаться на относящиеся к WSRF WSDL-файлы, которые находятся по адресу <http://www.globus.org/wsrf>.

### **1.4. Об авторе**

Бабу Сандарам (Babu Sundaram) является активным участником исследований по грид-компьютингу с момента появления пакета Globus Toolkit. Он входил в команду по реализации пакета Globus Toolkit во время своего пребывания в Argonne National Labs. Он имеет степень бакалавра в области mechanical engineering и степень магистра в области информатики. В университете Хьюстона (University of Huston) он продолжает работу над докторской диссертацией в области информатики. Автор опубликовал много работ в трудах различных конференций и семинаров, связанных с гридом. Является также соавтором ряда книг по грид-компьютингу. Увлекается преподаванием и иногда работает как лектор на курсах по Web-службам и различным аспектам информатики. Обратная связь приветствуется: [babu@cs.uh.edu](mailto:babu@cs.uh.edu).

## **2. Общий обзор**

### **2.1. О сущности грид-компьютинга**

Начнем с того, что сфера применения платформы WS-ресурса (WSRF) далеко перекрывает сферу применения грида, но именно в рамках грида WSRF была воплощена наиболее полно; поэтому грид является совершенным примером области, где возможности WSRF демонстрируются наиболее полно. Для тех, кто не знаком с гридом, просто расскажем, для чего он нужен.

Миллионы Web-пользователей принимали участие в обработке данных в рамках грида, большинство из них не имели об этом ни малейшего представления.

В 1998 году, когда проект Поиск Внеземного Разума (Search for Extraterrestrial Intelligence – SETI), оправившись от разрыва с NASA, реализовал свой новый план, был создан, возможно, самый известный проект в области грид-компьютинга. К этому времени SETI обладал невообразимо большим объемом данных, регулярно поставляемых радиотелескопом Arecibo, но не имел достаточной вычислительной мощности для их обработки. Для того чтобы справиться с этой проблемой, был создан специальный скринсейвер, который во время холостой работы компьютера запрашивал пакет данных с центрального сервера, обрабатывал его и результат отсылал обратно на сервер. Далее запрашивался новый пакет данных и цикл повторялся. Миллионы пользователей загрузили себе этот скринсейвер, приняв, таким образом, участие в этом проекте.

Более солидные грид-приложения выглядят значительно сложнее и ставят непростые задачи в отношении аутентичности, взаимодействия организационно разграниченных процессов, высокопроизводительных транспортных протоколов. И хотя энтузиастов грида уже тошнит от упоминания адреса SETI@Home, но он действительно является очень хорошим примером того, как работает грид. Здесь можно выделить следующие шаги:

1. Первоначальная задача разбивается на «единицы» так, чтобы было удобно решать задачу на многопроцессорных системах. Первоначальной задачей может быть вычислительная обработка, хранение данных или какой-нибудь другой тип обработки.

2. Множество полученных единиц распределяется среди клиентов; предполагается, что клиентов может быть очень много. Само распределение может совершаться по запросу клиентов, при организации пула клиентов, при выделении централизованного сервера, который распределяет работы среди свободных клиентов.

3. Общее наблюдение за прогрессом в решении первоначальной задачи поддерживается централизованной системой или группой систем.

Большинство грид-приложений в настоящее время следуют этой модели, когда централизованная система взаимодействует с системами, которые, как правило, географически удалены от центрального сервера.

## **2.2. Использование Web-служб: надежды и трудности**

Теперь, когда мы имеем дело с большим количеством клиентов, находящихся в разных местах, представляется вполне естественным, что в грид-компьютинге важное место должны были занимать Web-службы. Именно они предоставляют удобный, стандартизованный способ передавать информацию от одной системы к другой без обязательного обращения к зависящим от платформы или языка методам, таким как CORBA, DCOM или Java-RM.

Этого, однако, не случилось. Ранние грид-приложения использовали другие, менее портативные методы. Но почему?

Наиболее вероятной причиной была, пожалуй, архитектурная. Хотя грид-приложение может быть реализовано на нескольких машинах, оно по-прежнему является одним самостоятельным приложением; и концептуально, возможно, было трудно работать с архитектурой, не имеющей явно выраженного состояния.

Когда вы подсоединяетесь к базе данных с помощью клиента базы данных, вы к ней подсоединяетесь на самом деле и остаетесь подсоединенным во время добавления записи к базе и просмотра вновь добавленной записи. Любой другой

клиент, работающий с этой таблицей, добавленную вами запись видеть не будет до тех пор, пока вы не завершите транзакцию; но ваш сеанс база опознаёт и знает, что вы это вы

Web-службы работают не так. В случае Web-служб вы делаете запрос (например, на добавление записи в базу данных) и получаете ответ (например, подтверждения факта успешного добавления записи), затем вы отключаетесь. Нет необходимости поддерживать состояние сеанса связи. Аналогично HTTP – а в большинстве случаев Web-службы пользуются протоколом HTTP – каждый запрос не зависит от предыдущего запроса; Web-службы не имеют доступа к информации, не являющейся частью текущего входного сообщения, и она им и не требуется.

WSRF является попыткой решить указанную архитектурную проблему с помощью введения понятия «состояние», указав механизмы использования этого понятия.

### **2.3. Ресурсы с состоянием**

Итак, что в точности представляет собой «состояние», и что такое «ресурс с состоянием»?

Ресурс с состоянием это нечто, существующее даже тогда, когда вы с ним не взаимодействуете. Например, база данных существует, даже если вы не отправляете к ней никаких запросов. Находящийся на планетарной орбите спутник существует, даже если он не является предметом вашего разговора. Даже переменная для простого счетчика должна существовать в промежутках времени между ее обновлениями; иначе она будет возвращать одно и то же значение.

Но еще более важно осознать, что понятие состояния содержит в себе также идею свойств. Когда вас просят вернуть некоторый объект в том же самом состоянии, в котором вы его одолжили, то вам необходимо восстановить значения некоторых свойств таких, как уровень чистоты, показатель работоспособности, объем горючего в бачке. То же самое справедливо и для ресурсов с состоянием; они обладают свойствами, характеризующими их состояние; и наше взаимодействие с ними осуществляется через посредство этих свойств, таких, например, как показано в разделе «Свойства WS-ресурса»

### **2.4. Web-службы + ресурсы с состоянием = WS-ресурсы**

В соответствии со спецификацией, WS-ресурс является объединением Web-службы и ресурса с состоянием, на который Web-служба воздействует. Но что это означает в действительности?

Посмотрим на это самым обыденным взглядом. Предположим, что мы имеем дело с системой, управляющей группой искусственных спутников. Каждый спутник является ресурсом с состоянием в том смысле, что он существует даже, когда мы о нем не думаем. У нас могла бы быть также Web-служба, обеспечивающая функционирование спутника в отношении его ориентации, сбора информации или даже корректировки высоты спутника.

Объединяя упомянутые ресурс и службу, мы создаем WS-ресурс. Заметим, что между ними необязательно должно быть взаимнооднозначное соответствие. Так, например, наша единственная Web-служба могла бы воздействовать на несколько различных спутников, создавая несколько различных WS-ресурсов,

относящихся к одной и той же Web-службе. С другой стороны, один единственный спутник мог бы взаимодействовать с несколькими различными службами, контролирующими, например, бортовые эксперименты или управляющие лазерным лучом для отражения неприятельской атаки. При этом создавались бы различные WS-ресурсы, относящиеся к одному и тому же ресурсу с состоянием.

Чтобы использовать WS-ресурс необходимо осознанно работать с его свойствами.

## **2.5. Свойства WS-ресурса**

Как сказано в разделе «Ресурсы с состоянием», ресурс с состоянием (и, следовательно, WS-ресурс) имеет различные, связанные с ним свойства. Спутник мог бы иметь, например, следующие свойства:

- широта (latitude)
- долгота (longitude)
- высота (altitude)
- угол наклона (pitch)
- угол поворота (roll)
- отклонение от курса (yaw)
- расстояние до фокуса (focalLength)
- текущий обзор (currentView)

В нашем случае первые три свойства широта, долгота и высота определяют положение спутника. Следующие три свойства определяют его ориентацию, или направление его осмотра. Последние свойства определяют расстояние до точки фокусировки спутника и объекты, находящиеся в фокусе.

Значения этих свойств определяют состояние ресурса. Измените значение свойств – и вы измените его состояние. Именно таким образом мы и собираемся работать со спутником. Для изменения его положения мы изменим одно из позиционных свойств. Для изменения его ориентации мы изменим соответственно одно из ориентационных свойств. В общем, все, что мы захотим делать со спутником (в нашей ограниченной реализации), мы сможем сделать простым изменением соответствующих свойств.

Но как же мы это делаем?

## **2.6. Воспользуйтесь WSRF**

Итак, WS-ресурс является объединением ресурса с состоянием и Web-службы; наша работа с ним заключается в проверке и модификации его свойств.

Делать это можно многими различными способами, вопрос – каким конкретно. Что нам было нужно – это стандартный способ делать запросы для получения и изменения различных свойств

Воспользуемся WS-ресурс инфраструктурой (WSRF). WSRF – это ряд спецификаций, которые определяют стандартные «шаблоны сообщений», или, иначе, способы запроса значений свойства или свойств или способы указания того, что эти свойства должны быть изменены. Более того, WSRF определяет стандартные способы разрешения всех различных проблемных вопросов работы с WS-ресурсами. Сюда входят простые задачи работы с отдельными свойствами, но

также и задачи их группировки для таких целей, как аутентификация для гарантии их своевременного уничтожения.

Эти операции WSRF определяет с помощью языковых конструкций Языка Описания Web-служб (WSDL). Созданный при этом WSDL-файл содержит сообщения, которыми обмениваются обе стороны диалога используемой Web-службы. Определяя WSDL-файл, WSRF определяет форму любого взаимодействия, которое происходит.

Когда мы говорим «WSRF», мы имеем в виду несколько самостоятельных спецификаций:

Спецификация *WS-ResourceProperties (WSRF-RP)* определяет форму, в которой свойства ресурса определяются в файле WSDL. Она определяет также форму сообщений, которые запрашивают и получают значения свойств ресурса; в ней объясняется так же, как изменять, добавлять и удалять свойства какого-либо WS-ресурса.

- В спецификации *WS-ResourceLifetime (WSRF-RL)* обсуждается ситуация, когда WS-ресурс должен прекратить свою активность или быть явно уничтожен, когда пропадает необходимость его существования.
- Спецификация *WS-ServiceGroup (WSRF-SG)* определяет способ создания набора Web-служб, такого, например, как регистр имеющихся сервисов.
- Спецификация *WS-Base Faults (WSRF-BF)* определяет стандартный способ фиксации ошибок в WSRF-приложении.
- вы должны заметить, что в этом списке нет спецификации, определяющей как все это должно работать совместно. Эту функцию выполняет (почти полностью) белый документ *Modeling stateful resources with Web services* (Моделирование ресурсов с состоянием посредством Web-служб), в котором объясняется общая концепция этих спецификаций и их связь друг с другом.

Кроме краткого описания некоторых из понятий, объясняемых в белом документе, в данном учебном пособии обсуждается спецификация СВОЙСТВА WS-РЕСУРСА. В следующих статьях этой серии будут обсуждены и другие WSRF-спецификации, также как и семейство спецификаций WS-уведомлений, на которые ссылаются повсюду в WSRF.

## 3. Что нужно знать о WSDL

### 3.1. Что такое WSDL, и зачем оно мне?

Прежде, чем перейти собственно к созданию WS-ресурса в файле WSDL, остановимся на минутку на том, для чего нужен WSDL-файл и какова структура содержащихся в нем описаний.

Web-службы – или, по крайней мере, Web-службы, относящиеся к WS-ресурсам – состоят из SOAP-сообщений. У SOAP-сообщения есть стандартный «конверт», в котором содержится «вложение». Вложение это данные, которые должны быть проанализированы сервером в случае запроса и клиентом в случае ответа. Рассмотрим следующее SOAP-сообщение:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <SetAltitudeRequest xmlns="http://example.com/satellite.xsd">
      <altitude>47700</altitude>
    </SetAltitudeRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Это сообщение содержит стандартный конверт в пространстве имен `http://schemas.xmlsoap.org/soap/envelope` (SOAP-ENV) и вложение в пространстве имен `http://example.com/satellite.xsd`

Вложение может быть чем угодно, и с этим связана следующая проблема: как определить то, что ваше приложение ожидает увидеть, и что оно возвратит? Вот именно для этого и нужен WSDL-файл. Мы обращаемся к WSDL-файлу главным образом для определения «шаблонов сообщений», которыми пользуется WS-ресурс. В этой главе мы рассмотрим только то, как отдельные части WSDL-файла согласуются друг с другом.

### **3.2. Сообщения и типы**

Начнем с того, что дадим определение вида реальных сообщений, которые мы собираемся посылать с клиента на сервер и с сервера клиенту:

```
<?xml version="1.0"?>
<definitions name="Satellite"
  targetNamespace="http://example.com/satellite.wsdl"
  xmlns:tns="http://example.com/satellite.wsdl"
  xmlns:satTypes="http://example.com/satellite.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
    <schema targetNamespace="http://example.com/satellite.xsd"
      xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="SetAltitudeRequest">
        <complexType>
          <all>
            <element name="altitude" type="float"/>
          </all>
        </complexType>
      </element>
      <element name="SetAltitudeResponse">
        <complexType>
          <all>
            <element name="result" type="string"/>
          </all>
        </complexType>
      </element>
    </schema>
  </types>

  <message name="SetAltitudeInput">
    <part name="body" element="satTypes:SetAltitudeRequest"/>
  </message>

  <message name="SetAltitudeOutput">
    <part name="body" element="satTypes:SetAltitudeResponse"/>
  </message>

</definitions>
```

Рассмотрим конец фрагмента текста на XML, здесь мы определили два типа сообщений. Первый тип, *SetAltitudeInput*, является типом сообщения, которое мы посылаем серверу как информацию ввода. Это вид тела SOAP-сообщения, которое вы видели в главе «Что такое WSDL, и зачем оно мне?» Второй тип сообщения, *SetAltitudeOutput*, это тот ответ, который сервер посылает клиенту. В обоих типах сообщений специфицируется элемент, который будет помещаться в тело SOAP-сообщения.

Точное определение этих элементов находится в схеме, находящейся в начале WSDL-файла. Например, сообщение *SetAltitudeInput* состоит из элемента *SetAltitudeRequest*, который содержит элемент *altitude*; причем значение элемента *altitude* должно быть типа *float*.

Теперь объединим эти сообщения и создадим операцию, которую должен выполнять сервер.

### 3.3. Типы портов и операции

Теперь, когда мы точно знаем сообщения, которые мы собираемся отослать серверу, нам необходимо специфицировать ту роль, которую они будут выполнять на сервере. Для этого мы создадим элемент *portType* (Тип порта) и связанные с ним *операции*:

```
...
<message name="SetAltitudeInput">
  <part name="body" element="satTypes:SetAltitudeRequest"/>
</message>

<message name="SetAltitudeOutput">
  <part name="body" element="satTypes:SetAltitudeResponse"/>
</message>

<portType name="AltitudePortType">
  <operation name="SetAltitude">
    <input message="tns:SetAltitudeInput"
      wsa:Action="http://example.com/SetAltitude" />
    <output message="tns:SetAltitudeOutput"
      wsa:Action="http://example.com/SetAltitudeResponse" />
  </operation>
</portType>

</definitions>
```

Теперь мы определили элемент *Тип порта* с именем *AltitudePortType*, с единственной *операцией* *SetAltitude*. На самом деле мы могли бы определить и связать с этим элементом любое количество операций, но пока, для простоты, мы не будем этого делать. В операции *SetAltitude* указаны два сообщения: одно сообщение ввода, *SetAltitudeInput*, и одно сообщение вывода, *SetAltitudeOutput*. (Мы поговорим об атрибуте *wsa:Action* в следующем параграфе. И обратите внимание на информацию пространства имен).

Вы можете также специфицировать «сообщение об ошибке», которое будет отсылаться в случае возникновения различных проблем; об этом мы будем рассказывать более подробно в дальнейших выпусках этой серии. Пока, как и прежде, не будем ничего усложнять.

### 3.4. Службы и связывание

До сих пор, мы пользовались элементом *Тип порта* для определения того, что может быть сделано, но не определили как это должно быть сделано. Для завершения нашего процесса определения нам необходимо создать элемент *binding* (*связывание*), где будет описано, как это должно быть сделано, и будет сделана привязка к фактической *службе* (*service*):

```
<portType name="AltitudePortType">
  <operation name="SetAltitude">
    <input message="tns:SetAltitudeInput"
      wsa:Action="http://example.com/SetAltitude" />
    <output message="tns:SetAltitudeOutput"

wsa:Action="http://example.com/SetAltitudeResponse" />
  </operation>
</portType>

  <binding name="AltitudeSoapBinding"
type="tns:AltitudePortType">
  <soap:binding style="document"

transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="SetAltitude">
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>

  <service name="SatelliteService">
    <port name="AltitudePort"
binding="tns:AltitudeSoapBinding">
      <soap:address location="http://example.com/satellite"/>
    </port>
  </service>

</definitions>
```

И снова, начнем с конца приведенного текста, с элемента *service*. В WSDL-файле может быть определено более одной службы. Например, у вас могли бы быть различные службы для различных целей, или различные службы для одной и той же цели в различных местах, или различные службы с различными связываниями, как, например, одна служба для связывания по SOAP, а одна – по SMTP.

В нашем случае, мы определяем одну службу *SatelliteService* с одним портом *AltitudePort*. Но что нам известно про этот порт? Что ж, мы знаем, что SOAP-запрос должен быть послано адресу *http://example.com/satellite*. Нам также известно, что за остальной исчерпывающей информацией о том, как посылать наши сообщения, мы должны обратиться к элементу *AltitudeSoapBinding*.

Элемент *AltitudeSoapBinding* говорит, что он является реализацией элемента *AltitudePortType*; отсюда следует, что мы знаем, *какие* сообщения можно посылать. Сам элемент *binding* определяет различные детали форматирования сообщений в каждой операции. В нашем случае мы используем стиль «документ/литеральный», что означает простое помещение определенных элементов в *Тело* сообщения.

И на этом мы завершаем выполнение поставленной задачи. Мы знаем, куда мы посылаем сообщения, как их нужно форматировать и как они должны

строиться. Когда мы начнем создавать WS-ресурс, мы вернемся к созданию конкретных сообщений, определенных с помощью WSRF, но сначала нам необходимо отвлечься на тему WS-адресации.

## 4. Что вам необходимо знать о WS-адресации

### 4.1. Что такое WS-адресация, и зачем она мне?

Раньше указать адрес Web-службы было очень легко. Что вам в действительности было необходимо – это соответствующий URL, вся остальная информация включалась в заголовок SoapAction или в само сообщение. Теперь же, когда приложения, использующие Web-службы, становятся все более и более сложными, это не всегда также просто. Что, если вы захотите, чтобы ответ посылался не первоначальному заказчику услуги, а куда-то в другое место? А что, если для определения искомого «места» потребуется и другая информация, такая, например, как идентификатор сеанса?

Или что нужно сделать, чтобы просто подсоединиться к некоторой конкретной реализации определенной Web-службы? С такой ситуацией мы столкнемся в случае WS-ресурса, и нам нужно уметь справляться с ней.

### 4.2. Рассмотрим WS-адресацию

WS-адресация предоставляет возможность специфицировать информацию о местоположении способом, отличным от простого указания Универсального Идентификатора Ресурса (URI) или URL. В нашем случае это по существу является стандартизованным способом добавлять различного вида информацию к SOAP-сообщению. Построим, например, такое SOAP-сообщение:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:wsa="http://www.w3.org/2005/02/addressing"
  xmlns:sat="http://example.org/satelliteSystem">
  <SOAP-ENV:Header>
    <wsa:To SOAP-
ENV:mustUnderstand="1">http://example.com/satellite</wsa:To>
    <wsa:Action>http://example.com/SetAltitude</wsa:Action>
    <sat:SatelliteId>SAT9928</sat:SatelliteId>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <SetAltitudeRequest
xmlns="http://example.com/satellite.xsd">
      <altitude>47700</altitude>
    </SetAltitudeRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Может показаться, что эту информацию необязательно приводить в реальном SOAP-сообщении. Но вспомните, что это сообщение может проходить через несколько различных систем на своем пути к цели своего назначения; и при этом могут быть использованы различные транспортные протоколы.

Чтобы специфицировать эту информацию, нам необходимо создать элемент *EndpointReference* (Ссылка на крайнюю точку).

### 4.3. Ссылки на крайнюю точку

WS-адресация вводит элемент *EndpointReference*. Этот элемент обеспечивает способ специфицировать информацию, необходимую для того, чтобы сообщение было доставлено в нужное место вместе с соответствующей ассоциированной информацией. Например, для сообщения, приведенного ранее, этот элемент будет иметь следующий вид:

```
<wsa:EndpointReference xmlns:wsa=
  "http://www.w3.org/2005/02/addressing"
  xmlns:sat="http://example.org/satelliteSystem">
  <wsa:Address>http://example.com/satellite</wsa:Address>
  <wsa:ReferenceProperties>
    <sat:SatelliteId>SAT9928</sat:SatelliteId>
  </wsa:ReferenceProperties>
</wsa:EndpointReference
```

Необходимо точно понимать связь между *EndpointReference* и самим SOAP-сообщением, потому что *EndpointReference* – это метод указания места конкретного WS-ресурса. Например, если мы запрашиваем создание нового WS-ресурса, то ответ состоит из *EndpointReference* и указывает на этот созданный WS-ресурс. Позднее мы встретимся с этим еще раз.

## 5. Создание WS-ресурса

### 5.1. Что такое WS-ресурс?

Теперь вам должна быть хорошо понятна идея WS-ресурса с концептуальной точки зрения, и вы умеете оперировать понятиями WSDL и WS-адресации. Можно начинать фактическое создание WS-ресурса, чтобы с ним работать в дальнейшем.

Начнем с определения того, чем является WS-ресурс на самом деле. В случае нашего примера со спутниками мы могли бы иметь различные типы WS-ресурсов, такие, как:

- служба, которая устанавливает и определяет высоту некоторого определенного спутника,
- служба, которая устанавливает и определяет местоположение и ориентацию некоторого определенного спутника,
- служба, которая обеспечивает доступ к процессу подсчета звезд в поле зрения некоторого определенного спутника.

В этом списке нужно обратить внимание на два момента: то, что все три WS-ресурса могут ссылаться на один и тот же спутник, и то, что WS-ресурс определяется комбинацией службы и ресурса с состоянием (спутника, в нашем случае), а количество операций, которые он может выполнять, в наше определение не входит.

## 5.2. Документ свойств ресурса

Теперь мы можем сказать, что WS-ресурс является объединением Web-службы и ресурса с состоянием, но мы не знаем, как представляется этот ресурс с состоянием в самом приложении. Ответ можно найти в документе *ResourceProperties (Свойства Ресурса)*. Как упомянуто в разделе «Ресурсы с состоянием», состояние объекта можно определить через значения его различных свойств. Поскольку это состояние объекта, которое нас интересует в действительности, мы можем представить наш ресурс с состоянием в виде XML-документа, в котором содержатся его свойства. Этот документ называется *Resource properties document*. В случае нашего спутника он мог бы представлять собой нечто вроде нижеследующего:

```
<satProp:GenericSatelliteProperties
xmlns:satProp="http://example.com/satellite">
  <satProp:latitude>30.3</satProp:latitude>
  <satProp:longitude>223.2</satProp:longitude>
  <satProp:altitude>47700</satProp:altitude>
  <satProp:pitch>49</satProp:pitch>
  <satProp:yaw>0</satProp:yaw>
  <satProp:roll>32</satProp:roll>
  <satProp:focalLength>21999992</satProp:focalLength>
  <satProp:currentView>
    http://example.com/satellite/2239992333.zip
  </satProp:currentView>
</satProp:GenericSatelliteProperties>
```

Изменение состояния спутника требуется изменения одного или нескольких из этих значений, и наоборот.

## 5.3. Переопределение WS-ресурса

Точно так же, как мы можем переопределить класс путем добавления членов и методов, мы можем расширить WS-ресурс путем добавления свойств. Например, рассмотрим ситуацию, в которой наш спутник занимается также подсчетом звезд. В добавление к обычным свойствам знакомого нам ресурса с состоянием у него может быть свойство *currentCount* (текущий Счетчик):

```
<satProp:GenericSatelliteProperties
  xmlns:satProp="http://example.com/satellite"
  xmlns:counterProp=
"http://example.com/satellite/CounterSatelliteProperties">
  <satProp:latitude>30.3</satProp:latitude>
  <satProp:longitude>223.2</satProp:longitude>
  <satProp:altitude>47700</satProp:altitude>
  <satProp:pitch>49</satProp:pitch>
  <satProp:yaw>0</satProp:yaw>
  <satProp:roll>32</satProp:roll>
  <satProp:focalLength>21999992</satProp:focalLength>
  <satProp:currentView>
    http://example.com/satellite/2239992333.zip
  </satProp:currentView>
  <counterProp:currentCount>92828</counterProp:currentCount>
</satProp:GenericSatelliteProperties>
```

Отметим, что новая информация относится к отдельному пространству имен.

#### **5.4. Объединение данной Web-службы с данным ресурсом: WSDL-файл**

К этому моменту мы создали модель нашего ресурса с состоянием (спутника), но чтобы завершить создание WS-ресурса, мы должны связать его с нашей службой, используя WSDL-файл.

Начнем с нашего основного WSDL-файла:

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions name="Satellite"
  targetNamespace="http://example.com/satellite"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:tns="http://example.com/satellite"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsrp=
"http://docs.oasis-open.org/wsrp/2004/06/wsrp-
WS-ResourceProperties-1.2-draft-01.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

  <wsdl:import namespace=
"http://docs.oasis-open.org/wsrp/2004/06/wsrp-
WS-ResourceProperties-1.2-draft-01.wsdl"
    location="WS-ResourceProperties.wsdl" />

  <types>
    <xsd:schema targetNamespace="http://example.com/satellite"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">

      <xsd:import namespace=
"http://schemas.xmlsoap.org/ws/2004/03/addressing"
        schemaLocation="WS-Addressing.xsd" />

    </xsd:schema>

  </types>
</definitions>
```

Сейчас он почти что пуст, но заметим, что для того, чтобы все это работало, необходимо еще импортировать два файла. Так как Вы, по-видимому, не создали на своей машине ссылочные файлы WS-ResourceProperties.wsdl и WS-Addressing.xsd, то, чтобы этим не заниматься, вы можете загрузить их упрощенные версии согласно описанию в разделе «Учебные ресурсы».

Теперь, когда у нас есть нужная нам общая схема WSDL-файла, начнем ее заполнять.

### **5.5. Добавление ресурса к WSDL-файлу**

Во-первых, добавим фактический ресурс с состоянием к нашему файлу и свяжем с ним Web-службу:

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions name="Satellite"
  targetNamespace="http://example.com/satellite"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:tns="http://example.com/satellite"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsrp=
"http://docs.oasis-open.org/wsrp/2004/06/wsrp-
WS-ResourceProperties-1.2-draft-01.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

  <wsdl:import namespace=
"http://docs.oasis-open.org/wsrp/2004/06/wsrp-
WS-ResourceProperties-1.2-draft-01.wsdl"
    location="WS-ResourceProperties.wsdl" />

  <types>
    <xsd:schema targetNamespace="http://example.com/satellite"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">

      <xsd:import namespace=
"http://schemas.xmlsoap.org/ws/2004/03/addressing"
        schemaLocation="WS-Addressing.xsd" />

      <xsd:element name="latitude" type="xsd:float" />
      <xsd:element name="longitude" type="xsd:float" />
      <xsd:element name="altitude" type="xsd:float" />
      <xsd:element name="pitch" type="xsd:float" />
      <xsd:element name="yaw" type="xsd:float" />
      <xsd:element name="roll" type="xsd:float" />
      <xsd:element name="focalLength" type="xsd:float" />
      <xsd:element name="currentView" type="xsd:string" />

      <xsd:element name="GenericSatelliteProperties">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="latitude" minOccurs="1"
              maxOccurs="1"/>
            <xsd:element ref="longitude" minOccurs="1"
              maxOccurs="1"/>
            <xsd:element ref="altitude" minOccurs="1"
              maxOccurs="1"/>
            <xsd:element ref="pitch" minOccurs="1"
              maxOccurs="1"/>
            <xsd:element ref="yaw" minOccurs="1"
              maxOccurs="1"/>
            <xsd:element ref="roll" minOccurs="1"
              maxOccurs="1"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
</definitions>
```

```
        <xsd:element ref="focalLength" minOccurs="1"
            maxOccurs="1"/>
        <xsd:element ref="currentView" minOccurs="1"
            maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>

</xsd:schema>

</types>

<portType name="SatellitePortType"
    wsrp:ResourceProperties=
    "tns:GenericSatelliteProperties">

</portType>

<binding name="SatelliteSoapBinding"
type="tns:SatellitePortType">
    <soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http"/>
</binding>

<service name="SatelliteService">
    <port name="SatellitePort"
        binding="tns:SatelliteSoapBinding">
        <soap:address location=
            "http://example.com/satellite"/>
    </port>
</service>

</definitions>
```

Мы начали с добавления базовых элементов Web-службы, элемента *service* и элемента *binding*, который ассоциирует *service* с элементом *portType*. Сам элемент *portType* пока что не содержит в себе никаких операций, самой главной составляющей в нем является атрибут *wsrp:ResourceProperties*. Этот атрибут говорит о том, что любая операция, которую выполняет наша Web-служба, совершается над определенным типом ресурса с состоянием, так, как это определено элементом *GenericSatelliteProperties*. Элемент *GenericSatelliteProperties* определен в элементе *schema*. Объединение этого ресурса с состоянием и нашей Web-службы является требуемым WS-ресурсом.

Отметим, что в соответствии со спецификацией, при создании документа свойств ресурса (в нашем случае такого, как *GenericSatelliteProperties*), вы должны использовать именно такой стиль, как здесь, когда происходят ссылки на уже созданные фактические элементы, а не происходит их создание в режиме *inline*.

Теперь добавим несколько настоящих операций в базовые элементы WSDL-файлу, и посмотрим, как это все работает.

## 5.6. Запрос нового спутника

Конечная цель нашего упражнения заключается, разумеется, в том, чтобы выполнить реальные операции над нашим WS-ресурсом; поэтому первое, что мы сделаем, это создадим ссылку на фактический экземпляр WS-ресурса:

```
...
  <types>
    <xsd:schema targetNamespace="http://example.com/satellite"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:import namespace=
        "http://schemas.xmlsoap.org/ws/2004/03/addressing"
        schemaLocation="WS-Addressing.xsd" />
      <xsd:element name="createSatellite">
        <xsd:complexType/>
      </xsd:element>
      <xsd:element name="createSatelliteResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="wsa:EndpointReference"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="GenericSatelliteProperties">
        ...
      </xsd:element>
    </xsd:schema>
  </types>
  <message name="CreateSatelliteRequest">
    <part name="request" element="tns:createSatellite"/>
  </message>
  <message name="CreateSatelliteResponse">
    <part name="response" element=
      "tns:createSatelliteResponse"/>
  </message>
  <portType name="SatellitePortType"
    wsrp:ResourceProperties=
      "tns:GenericSatelliteProperties">
    <operation name="createSatellite">
      <input message="tns:CreateSatelliteRequest"
        wsa:Action=
          "http://example.com/CreateSatellite" />
      <output message="tns:CreateSatelliteResponse"
        wsa:Action=
          "http://example.com/CreateSatelliteResponse" />
    </operation>
  </portType>
</binding>
</service>
```

```
</portType>

<binding name="SatelliteSoapBinding" type=
"tns:SatellitePortType">
  <soap:binding style="document" transport=
"http://schemas.xmlsoap.org/soap/http"/>
    <operation name="createSatellite">
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>

  <service name="SatelliteService">
    <port name="SatellitePort" binding=
"tns:SatelliteSoapBinding">
      <soap:address location=
"http://example.com/satellite"/>
    </port>
  </service>

</definitions>
```

На первый взгляд, этот файл не очень сильно отличается от WSDL-файла, который мы создали в главе «Что вам нужно знать о WSDL». У нас есть *service*, который указывает на элемент *binding*, в котором объясняется, как нужно реализовать *portType*. В элементе *portType* определяется одна операция *createSatellite*, использующая *input* и *output* сообщения. Структура этих сообщений определена в элементе *schema*.

Но, говоря об этом файле, нужно обратить внимание на одну вещь, которая несколько отличает его от первоначального. Вместо того, чтобы возвращать простое значение, служба возвращает *EndpointReference*, где содержится ссылка на вновь созданный WS-ресурс. Посмотрим, как это используется в SOAP-сообщении.

## 5.7. SOAP-запрос

Фактический SOAP-запрос на создание WS-ресурса выглядит очень просто:

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <createSatellite xmlns=
"http://example.com/satellite"/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

У нас еще нет фактического объекта, поэтому сам запрос направляется согласно URI, записанному в WSDL-файле, и мы определили содержание запроса как простой элемент *createSatellite*.

Ответ несколько интереснее

## 5.8. SOAP-ответ

После того, как вы послали запрос на создание нового спутника, сервер создает ссылку на новый WS-ресурс и отправляет ее обратно в форме *EndpointReference*:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <wsa:EndpointReference
      xmlns:wsa="http://www.w3.org/2005/02/addressing"
      xmlns:sat="http://example.org/satelliteSystem">
      <wsa:Address>
        http://example.com/satellite</
        wsa:Address>
      <wsa:ReferenceProperties>
        <sat:SatelliteId>
          SAT9928</sat:
          SatelliteId>
        </wsa:ReferenceProperties>
      </wsa:EndpointReference>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Заметим, что элемент *Address* в *EndpointReference* указывает на тот же самый URI, который мы поместили в WSDL-файл, поэтому информация отправляется, как и раньше, по прежнему адресу; только информации стало больше.

Но мы должны также отметить, что мы имеем дело не с обычной конечной ссылкой. В элементе *EndpointReference* указан идентификатор, который обязательно должен использоваться при идентификации WS-ресурса. В связи с этим, этот элемент можно рассматривать как действительно квалифицированную конечную ссылку на WS-ресурс. Мы можем использовать эту информацию при последующих обращениях к WS-ресурсу, чем мы воспользуемся в дальнейшем.

## 6. Как работать с WS-ресурсом: доступ к свойствам

### 6.1. Что мы пытаемся сделать

Вот мы создали WS-ресурс, что мы теперь можем с его помощью сделать?

Грубо говоря, все то, что мы можем сделать, изменяя его свойства. Например, мы могли бы изменить размер орбиты спутника, изменив его свойство *altitude* (высота). Не совсем так, простое изменение значения высоты на движение спутника влияния не оказывает; фактическое перемещение спутника возлагается на программное обеспечение, которое поддерживает нашу Web-службу. Но именно в этом и заключается смысл создания WS-ресурса, чтобы предоставить способ манипулирования с объектами путем изменения их свойств. В спецификации просто объясняется, как вы можете сообщить Web-службе об этих

изменениях. Для спецификации неважно, как приложение физически манипулирует объектом; для нас, впрочем, тоже.

Но прежде, чем мы начнем задавать команды по изменению свойств, посмотрим внимательнее на сами эти свойства. В этом разделе мы сначала рассмотрим свойство *altitude* для спутникового WS-ресурса, который мы создали в главе «Создание WS-ресурса». Затем мы перейдем к одновременному запросу всех ориентационных свойств. И затем покажем, как можно использовать выражения XPath для выбора множественных значений.

## 6.2. Выбор значения свойства

Запросить значение свойства – это просто построить соответствующее SOAP-сообщение. Например, предположим, что мы хотим получить значение свойства *altitude*. Базовое SOAP-сообщение будет выглядеть примерно так:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:wsrp=
"http://docs.oasis-open.org/wsrp/2004/06/wsrp-WS-ResourceProperties-1.2-draft-01.xsd">
  <SOAP-ENV:Header>...</SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <wsrp:GetResourceProperty xmlns:satProp=
      "http://example.com/satellite">
      satProp:altitude
    </wsrp:GetResourceProperty>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Элемент *wsrp:GetResourceProperty* является частью спецификации WS-ResourceProperty. Нам даже не нужно его определять в WSDL-файле. Он указывает нам место, где нужно специфицировать то свойство, для которого мы хотим получить возвращаемое значение.

Впрочем, построение SOAP-сообщения еще не завершено. Да, это SOAP-сообщение, но если мы отошлем его Web-службе в приведенном выше виде, то служба не будет знать на какой WS-ресурс мы ссылаемся. Сейчас мы займемся решением этого вопроса.

## 6.3. Полный SOAP-запрос

В предыдущем разделе «Выбор значения свойства», мы создали SOAP-сообщение, в котором указывалось нужное нам свойство, но для какого WS-ресурса?

Когда мы создали спутник, Web-служба возвратила нам ссылку на крайнюю точку, указывающую на вновь созданный WS-ресурс. Мы можем взять эту информацию и добавить ее в заголовок SOAP-сообщения, например, следующим образом:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:sat="http://example.org/satelliteSystem"
  xmlns:wsa="http://www.w3.org/2005/02/addressing"
  xmlns:wsrp="http://docs.oasis-open.org/wsrp/2004/06/wsrp-WS-ResourceProperties-1.2-draft-01.xsd">
  <SOAP-ENV:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsrp/2004/06/WS-ResourceProperties/GetResourceProperty
    </wsa:Action>
    <wsa:To SOAP-ENV:mustUnderstand="1">
      http://example.com/satellite
    </wsa:To>
    <sat:SatelliteId>SAT9928</sat:SatelliteId>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <wsrp:GetResourceProperty
      xmlns:satProp="http://example.com/satellite">
      satProp:altitude
    </wsrp:GetResourceProperty>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Элемент *wsa:Action* не является частью созданной ссылки на крайнюю точку; он изменяется в зависимости от того, что мы пытаемся сделать. В рассматриваемом случае мы используем действие *GetResourceProperty*. Элемент *wsa:To* выбирает значение по адресу *wsa:Address* (из ссылки на крайнюю точку), и любые значения *wsa:ReferencesProperty* непосредственно переносятся в заголовок (*Header*).

Вы заметили, что мы не обсуждали значение *SatelliteId*. Это было сделано специально. Любая, содержащаяся в ссылке на крайнюю точку информация, относящаяся к идентификации конкретного WS-ресурса, должна игнорироваться вашим приложением, вы просто передаете ее при отправлении сообщений. В соответствии со спецификацией, считается некорректным даже попытка интерпретации значения *SatelliteId*. Безусловным является предположение, что оно передается как «черный ящик», ведущий независимую и недоступную для наблюдений жизнь.

#### 6.4. Получение свойства ресурса

После того, как вы запросили свойство, вы ожидаете, когда вам будет возвращено его значение, и спецификация свойства WS-ресурса определяет форму ожидаемого сообщения. В нашем случае мы бы получили сообщение примерно такого вида:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
"http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:sat="http://example.org/satelliteSystem"
  xmlns:wsa="http://www.w3.org/2005/02/addressing"
  xmlns:wsrp=
"http://docs.oasis-open.org/wsrp/2004/06/wsrp-WS-
ResourceProperties-1.2-draft-01.xsd">
  <SOAP-ENV:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsrp/2004/06/WS-
      ResourceProperties/GetResourcePropertyResponse
    </wsa:Action>
    <wsa:To SOAP-ENV:mustUnderstand="1">
      http://example.com/myClient
    </wsa:To>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <wsrp:GetResourcePropertyResponse
      xmlns:satProp=
      "http://example.com/satellite">
      <satProp:altitude>
        47700
      </
      satProp:altitude>
    </wsrp:GetResourcePropertyResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Здесь информация заголовка Header относится не к службе, а к клиенту; `http://example.com/myClient` – это URI клиента, который должен получить ответ. В большинстве случаев это тот же самый клиент, который сделал запрос, но если вы хотите отправить ответ куда-нибудь в другое место, вы можете воспользоваться элементом `wsa:Reply-To`.

Что касается самого тела сообщения, где у нас также имеется стандартный элемент `wsrp:GetResourcePropertyResponse`, то теперь в нем содержится фактическое затребованное свойство с его текущим значением.

Посмотрим теперь, как это выглядит в WSDL-файле.

## 6.5. WSDL-файл

Чтобы добавить эти возможности к нашему приложению, нам нужно добавить их к WSDL-файлу. Но так как мы используем стандартные шаблоны обмена сообщениями, которые уже были определены, нам нужно всего лишь добавить новую операцию; например, так:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="Satellite"
  targetNamespace="http://example.com/satellite"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:tns="http://example.com/satellite"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing"
  xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsrp="http://docs.oasis-open.org/wsrp/2004/06/
wsrf-WS-ResourceProperties-1.2-draft-01.xsd"
  xmlns:wsrpwsdl=
"http://docs.oasis-open.org/wsrp/2004/06/wsrp-WS-
ResourceProperties-1.2-draft-01.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

  <wSDL:import namespace="http://docs.oasis-open.org/wsr
f/2004/0
6/wsrp-WS-ResourceProperties-1.2-draft-01.wsdl"
    location="WS-ResourceProperties.wsdl" />

  <types>
    <xsd:schema targetNamespace="http://example.com/satellite"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    ...
    </xsd:schema>

  </types>

  <message name="CreateSatelliteRequest">
    <part name="request" element="tns:createSatellite"/>
  </message>

  <message name="CreateSatelliteResponse">
    <part name="response" element="tns:createSatelliteResponse"/>
  </message>

  <portType name="SatellitePortType"
wsrp:ResourceProperties="tns:GenericSatelliteProperties">

    <operation name="createSatellite">
      <input message="tns:CreateSatelliteRequest"
        wsa:Action="http://example.com/CreateSatellite" />
```

```

        <output message="tns:CreateSatelliteResponse"

wsa:Action="http://example.com/CreateSatelliteResponse" />
        </operation>

        <operation name="getAltitude">
            <input message="wsrpwsdl:GetResourcePropertyRequest"
                wsa:Action="http://docs.oasis-open.org/wsrf/2004/
                06/WS-ResourceProperties/GetResourceProperty/">
                <output message="wsrpwsdl:GetResourcePropertyResponse"
                    wsa:Action="http://docs.oasis-open.org/wsrf/2004/
                    06/WS-
ResourceProperties/GetResourcePropertyResponse/">
            </operation>
        </portType>

        <binding name="SatelliteSoapBinding"
type="tns:SatellitePortType">
            <soap:binding style="document"
                transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="createSatellite">
                <input>
                    <soap:body use="literal"/>
                </input>
                <output>
                    <soap:body use="literal"/>
                </output>
            </operation>
            <operation name="getAltitude">
                <input>
                    <soap:body use="literal"/>
                </input>
                <output>
                    <soap:body use="literal"/>
                </output>
            </operation>
        </binding>

        <service name="SatelliteService">
            <port name="SatellitePort" binding=
                "tns:SatelliteSoapBinding">
                <soap:address location=
                    "http://example.com/satellite"/>
            </port>
        </service>

    </definitions>

```

Сначала в элементе *portType*, мы создаем новую операцию с именем *getAltitude*. Эта операция содержит два сообщения *input* и *output*, но оба эти сообщения были уже определены в файле *WS-ReferenceProperties.wsdl*, который мы импортировали ранее, поэтому все, что нам нужно сделать – это сослаться на них, используя подходящий псевдоним пространства имен.

После того, как мы создали требуемую операцию, нам нужно просто добавить ее к связыванию. Теперь мы можем идти дальше.

## 6.6. Запрос множества свойств

К счастью, мы не обязаны выбирать значения только одного свойства. Мы можем также выбирать множество свойств:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:sat="http://example.org/satelliteSystem"
  xmlns:wsa="http://www.w3.org/2005/02/addressing"
  xmlns:wsrp="http://docs.oasis-open.org/wsrp/2004/06/wsrp-
WS-ResourceProperties-1.2-draft-01.xsd">
  <SOAP-ENV:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsrp/2004/06/
      WS-ResourceProperties/GetMultipleResourceProperties
    </wsa:Action>
    <wsa:To SOAP-ENV:mustUnderstand="1">
      http://example.com/satellite
    </wsa:To>
    <sat:SatelliteId>SAT9928</sat:SatelliteId>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <wsrp:GetMultipleResourceProperties
      xmlns:satProp="http://example.com/satellite">
      <wsrp:ResourceProperty>
        satProp:roll
      </wsrp:ResourceProperty>
      <wsrp:ResourceProperty>
        satProp:pitch
      </wsrp:ResourceProperty>
      <wsrp:ResourceProperty>
        satProp:yaw
      </wsrp:ResourceProperty>
    </wsrp:GetMultipleResourceProperties>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Как и прежде, информация в элементе Header имеет своим источником ссылку на крайнюю точку.

## 6.7. Получение множества свойств

Ответное сообщение здесь подобно своему аналогу для случая одного свойства:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
"http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:sat="http://example.org/satelliteSystem"
  xmlns:wsa="http://www.w3.org/2005/02/addressing"
  xmlns:wsrp="http://docs.oasis-open.org/wsrp/200
4/06/wsrp-WS-ResourceProperties-1.2-draft-01.xsd">
  <SOAP-ENV:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsrp/2004/06/WS-Res
      ourceProperties/GetMultipleResourcePropertiesResponse
    </wsa:Action>
    <wsa:To SOAP-ENV:mustUnderstand="1">
      http://example.com/myClient
    </wsa:To>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <wsrp:GetMultipleResourcePropertiesResponse
      xmlns:satProp="http://example.com/satellite">
      <satProp:roll>32</satProp:roll>
      <satProp:pitch>49</satProp:pitch>
      <satProp:yaw>0</satProp:yaw>
    </wsrp:GetMultipleResourcePropertiesResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Опять же, добавим этот фрагмент к WSDL-файлу.

## 6.8. WSDL-файл

Так как фактически передаваемые сообщения были ранее определены в файле *WS-ResourceProperties.wsdl*, то, как и прежде, для добавления этой возможности к приложению, нам необходимо всего лишь создать новый элемент *operation*.

```

...
    <message name="CreateSatelliteRequest">
      <part name="request" element=
        "tns:createSatellite"/>
    </message>

    <message name="CreateSatelliteResponse">
      <part name="response" element=
        "tns:createSatelliteResponse"/>
    </message>

    <portType name="SatellitePortType"
      wsrp:ResourceProperties =
        "tns:GenericSatelliteProperties">

      <operation name="createSatellite">
        <input message="tns:CreateSatelliteRequest"
          wsa:Action=
            "http://example.com/CreateSatellite" />
        <output message="tns:CreateSatelliteResponse"
          wsa:Action=
            "http://example.com/CreateSatelliteResponse" />
      </operation>

      <operation name="getAltitude">
        <input message="wsrpwsdl:
GetResourcePropertyRequest"
          wsa:Action="http://docs.oasis-open.org/wsrfl/
2004/06/WS-ResourceProperties/GetResourceProperty"/>
        <output message=
          "wsrpwsdl:GetResourcePropertyResponse"
          wsa:Action="http://docs.oasis-open.org/wsrfl/2004/0
6/WS-ResourceProperties/GetResourcePropertyRespo
nse"/>
      </operation>

      <operation name="getOrientation">
        <input message="wsrpwsdl:
GetMultipleResourcePropertiesRequest"
          wsa:Action="http://docs.oasis-open.org/wsrfl/2004/0
6/WS-ResourceProperties/GetMultipleResourceProper
ties"/>

        <output message=
          "wsrpwsdl:GetMultipleResourcePropertiesResponse"
          wsa:Action=
            "http://docs.oasis-open.org/wsrfl/2004/06/WS-
Resource
Properties/GetMultipleResourcePropertiesResponse"/>
      </operation>

    </portType>

```

```
<binding name="SatelliteSoapBinding" type=
  "tns:SatellitePortType">
  <soap:binding style="document"
    transport=
      "http://schemas.xmlsoap.org/soap/http"/>
  <operation name="createSatellite">
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
  <operation name="getAltitude">
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
  <operation name="getOrientation">
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>

  <service name="SatelliteService">
    <port name="SatellitePort" binding="tns:SatelliteSoapBinding">
      <soap:address location="http://example.com/satellite"/>
    </port>
  </service>

</definitions>
```

После того, как мы добавили операцию *getOrientation* к *binding*, она может использоваться в приложении.

Кроме того, мы можем производить поиск свойств в различных сочетаниях, что мы сейчас и увидим.

## 6.9. Использование для запросов выражений XPath

Хотя выполнять запросы желаемых свойств достаточно просто, могут возникнуть ситуации, когда придется прибегнуть к несколько иной схеме запросов. Вместо того, чтобы запросить свойство ресурса по имени, мы можем использовать запрос в виде конструкций языка XPath. (Более подробно с XPath вы можете познакомиться по ссылкам в разделе «Учебные ресурсы».) Например, если мы не знаем наверняка, существует ли данное конкретное свойство, то могли бы применить к этому свойству соответствующую функцию XPath.

Другой полезной возможностью, которую предоставляет XPath, является возможность запрашивать более одного одинаково поименованного свойства (чем мы воспользуемся во второй части нашей серии) или даже использовать в запросах значения параметров (чем мы воспользуемся здесь). Например, мы могли бы воспользоваться выражением XPath в функции `boolean()` для определения того, что наш спутник ориентирован правильно, не прибегая к явному анализу соответствующих данных:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
"http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:sat="http://example.org/satelliteSystem"
  xmlns:wsa="http://www.w3.org/2005/02/addressing"
  xmlns:wsrp="http://docs.oasis-open.org/wsrf/20
04/06/wsrf-WS-ResourceProperties-1.2-draft-01.xsd">
  <SOAP-ENV:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsrf/2004/06/WS-ResourcePro
      perties/QueryResourceProperties
    </wsa:Action>
    <wsa:To SOAP-ENV:mustUnderstand="1">
      http://example.com/satellite
    </wsa:To>
    <sat:SatelliteId>SAT9928</sat:SatelliteId>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <wsrp:QueryResourceProperties>
      <wsrp:QueryExpression Dialect=
        "http://www.w3.org/TR/1999/REC-xpath-19991116">
        boolean(/*/pitch=25 and */roll=0 and */yaw=10)
      </wsrp:QueryExpression>
    </wsrp:QueryResourceProperties>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Обратите внимание на то, как мы использовали атрибут `Dialect` для указания используемой у нас версии XPath V1.0 вместо версии XPath V2.0 (<http://www.w3.org/TR/2003/WD-xpath20-20031112>).

Спецификация не ограничивает диалекты, которые вы можете поддерживать, но если реализация не опознает используемый `Dialect`, она должна возвращать `fault`, или ошибку.

## 6.10. Результаты запроса

Наши результаты выглядят, в общем, так же, как два предыдущих ответа:

```

<SOAP-ENV:Envelope xmlns:SOAP-ENV=
"http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:sat="http://example.org/satelliteSystem"
  xmlns:wsa="http://www.w3.org/2005/02/addressing"
  xmlns:wsrp="http://docs.oasis-open.org/wsrp/20
04/06/wsrp-WS-ResourceProperties-1.2-draft-01.xsd">
  <SOAP-ENV:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsrp/2004/06/
      WS-ResourceProperties/Quer
yResourcePropertiesResponse
    </wsa:Action>
    <wsa:To SOAP-ENV:mustUnderstand="1">
      http://example.com/myClient
    </wsa:To>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <wsrp:QueryResourcePropertiesResponse>
      false
    </wsrp:QueryResourcePropertiesResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

В нашем случае мы просто возвращаем булевское значение, но можно было бы вернуть значение любого типа из тех, что возвращает XPath.

## 6.11. WSDL-файл

И опять же добавим к WSDL файлу новый элемент *operation*:

```

...
  <portType name="SatellitePortType"
    wsrp:ResourceProperties=
      "tns:GenericSatelliteProperties">
...
    <operation name="getOrientation">
      <input message=
        "wsrpwsdl:GetMultipleResourcePropertiesRequest"
        wsa:Action="http://docs.oasis-open.org/wsrp/20
04/06/WS-
ResourceProperties/GetMultipleResourceProperties"/>
      <output message=
        "wsrpwsdl:GetMultipleResourcePropertiesResponse"
        wsa:Action="http://docs.oasis-open.org/wsrp/2004/06/W
S-
ResourceProperties/GetMultipleResourcePropertiesResponse"/>
    </operation>

```

```

    <operation name="checkOrientation">
      <input message=
        "wsrpwsdl:QueryResourcePropertiesRequest"
        wsa:Action="http://docs.oasis-open.org/wsr/20
          04/06/WS-
ResourceProperties/QueryResourceProperties"/>
      <output message=
        "wsrpwsdl:QueryResourcePropertiesResponse"
        wsa:Action="http://docs.oasis-
open.org/wsr/2004/06/W
          S-
ResourceProperties/QueryResourcePropertiesResponse"/>
    </operation>

  </portType>

  <binding name="SatelliteSoapBinding" type=
    "tns:SatellitePortType">
    <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
  ...
  <operation name="getOrientation">
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
  <operation name="checkOrientation">
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>

<service name="SatelliteService">
  <port name="SatellitePort" binding=
    "tns:SatelliteSoapBinding">
    <soap:address location=
      "http://example.com/satellite"/>
  </port>
</service>

</definitions>

```

С выбором значений свойств мы полностью разобрались. Теперь давайте перейдем к заданию этим свойствам значений.

## 7. Как работать с WS-ресурсом: установка значений СВОЙСТВ

### 7.1. Что мы пытаемся сделать

К настоящему моменту мы создали WS-ресурс, и мы сделали несколько синтаксически различных проверок свойств, представляющих его состояние. Все это, впрочем, не имеет большого значения в отношении действительного манипулирования WS-ресурсом. В этом разделе мы рассмотрим вопросы добавления, дезактивации и удаления свойств WS-ресурса.

До сих пор наши спутники находились в небе на синхронизированной с землей орбите в стационарном режиме. Никуда конкретно они не были ориентированы. В этом разделе мы добавим *ResourceProperty*, соотносящееся с некоторой целью. Мы затем переместим наш спутник в направлении этой цели, обновив позиционные свойства; и затем мы удалим добавленное свойство, относящееся. И, наконец, мы добавим соответствующие операции к WSDL-файлу.

### 7.2. Добавление свойства

При добавлении свойства к WS-ресурсу мы используем элемент *Insert*:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
"http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:sat="http://example.org/satelliteSystem"
  xmlns:wsa="http://www.w3.org/2005/02/addressing"
  xmlns:wsrp="http://docs.oasis-open.org/wsrfl/
2004/06/wsrfl-WS-ResourceProperties-1.2-draft-01.xsd">
  <SOAP-ENV:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsrfl/2004/06/W
      S-ResourceProperties/SetResourceProperties
    </wsa:Action>
    <wsa:To SOAP-ENV:mustUnderstand="1">
      http://example.com/satellite
    </wsa:To>
    <sat:SatelliteId>SAT9928</sat:SatelliteId>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <wsrp:SetResourceProperties
      xmlns:satProp="http://example.com/satellite">

      <wsrp:Insert>
        <satProp:targetCoords>
          36n11, 115w08
        </satProp:targetCoords>
      </wsrp:Insert>

    </wsrp:SetResourceProperties>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Новому свойству мы можем присвоить произвольное имя, но мы должны позаботиться, чтобы этот новый элемент был разрешен в документе *Resource*

*Properties*. (Мы это сделаем, когда будем актуализировать WSDL-файл в разделе «WSDL-файл».)

### 7.3. Результат добавления свойства

Когда действие по добавлению, удалению или изменению свойства выполняется успешно, мы получаем сообщение-ответ, которое просто фиксирует выполнение этого действия:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
"http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:sat="http://example.org/satelliteSystem"
  xmlns:wsa="http://www.w3.org/2005/02/addressing"
  xmlns:wsrp="http://docs.oasis-open.org/wsrp/
2004/06/wsrp-WS-ResourceProperties-1.2-draft-01.xsd">
  <SOAP-ENV:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsrp/2004/06/W
S-ResourceProperties/SetResourcePropertiesResponse
    </wsa:Action>
    <wsa:To SOAP-ENV:mustUnderstand="1">
      http://example.com/myClient
    </wsa:To>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <wsrp:SetResourcePropertiesResponse>
    </wsrp:SetResourcePropertiesResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Этот ответ идентичен ответу в случае действий *Update* и *Delete*.

### 7.4. Изменение значения свойства

В главе «Добавление свойства» мы добавили новое свойство, но мы можем также изменять значения существующих свойств. Например, мы можем приказать системе переместить спутник путем изменения позиционных свойств с помощью элемента *Update*:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
"http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:sat="http://example.org/satelliteSystem"
  xmlns:wsa="http://www.w3.org/2005/02/addressing"
  xmlns:wsrp="http://docs.oasis-open.org/wsrp/2004/0
6/wsrp-WS-ResourceProperties-1.2-draft-01.xsd">
  <SOAP-ENV:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsrp/2004/06/WS-Re
sourceProperties/SetResourceProperties
    </wsa:Action>
    <wsa:To SOAP-ENV:mustUnderstand="1">
      http://example.com/satellite
    </wsa:To>
    <sat:SatelliteId>SAT9928</sat:SatelliteId>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <wsrp:SetResourceProperties
      xmlns:satProp="http://example.com/satellite">

      <wsrp:Update>
        <satProp:latitude>36.11</satProp:latitude>
      </wsrp:Update>

      <wsrp:Update>
        <satProp:longitude>158.08</satProp:latitude>
      </wsrp:Update>

    </wsrp:SetResourceProperties>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

В этом случае мы используем две компоненты *Update*, но вообще мы можем использовать любую комбинацию компонент *Insert*, *Update* и *Delete*.

## 7.5. Удаление свойства

Свойство может быть удалено полностью. Например, если бы мы решили больше не быть ориентированными на конкретную цель, мы могли бы удалить соответствующее свойство:

```

<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:sat="http://example.org/satelliteSystem"
      xmlns:wsa="http://www.w3.org/2005/02/addressing"
      xmlns:wsrp="http://docs.oasis-open.org/wsrp/2004/06/wsrp-
WS-ResourceProperties-1.2-draft-01.xsd">
  <SOAP-ENV:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsrp/2004/06/WS-Resou
rceProperties/SetResourceProperties
    </wsa:Action>
    <wsa:To SOAP-ENV:mustUnderstand="1">
      http://example.com/satellite
    </wsa:To>
    <sat:SatelliteId>SAT9928</sat:SatelliteId>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <wsrp:SetResourceProperties
      xmlns:satProp="http://example.com/satellite">

      <wsrp>Delete resourceProperty="targetCoords"/>

    </wsrp:SetResourceProperties>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Опять-таки, абсолютно необходимо, чтобы определение схемы для нашего документа свойств ресурса разрешило нам делать это изменение.

## 7.6. WSDL-файл

Здесь также мы можем просто добавить новую операцию, так как стандартные элементы были уже определены в файле WS-ResourceProperties.wsdl:

```

<?xml version="1.0" encoding="UTF-8"?>
<definitions name="Satellite"
  targetNamespace="http://example.com/satellite"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:tns="http://example.com/satellite"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing"
  xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsrp="http://docs.oasis-open.org/wsrp/2004/06/
wsrp-WS-ResourceProperties-1.2-draft-01.xsd"
  xmlns:wsrpwsdl="http://docs.oasis-open.org/wsrp/200
4/06/wsrp-WS-ResourceProperties-1.2-draft-01.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

  <wsdl:import namespace="http://docs.oasis-open.org/
wsrp/2004/06/wsrp-WS-ResourceProperties-1.2-draft-01.wsdl"
    location="WS-ResourceProperties.wsdl" />

```

```
<types>
  <xsd:schema
    targetNamespace="http://example.com/satellite"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:import namespace=
"http://schemas.xmlsoap.org/ws/2004/03/addressing"
    schemaLocation="WS-Addressing.xsd" />

    <xsd:element name="createSatellite">
      <xsd:complexType/>
    </xsd:element>

    <xsd:element name="createSatelliteResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref=
            "wsa:EndpointReference" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="latitude" type="xsd:float" />
    <xsd:element name="longitude" type="xsd:float" />
    <xsd:element name="altitude" type="xsd:float" />
    <xsd:element name="pitch" type="xsd:float" />
    <xsd:element name="yaw" type="xsd:float" />
    <xsd:element name="roll" type="xsd:float" />
    <xsd:element name="focalLength" type="xsd:float" />
    <xsd:element name="currentView" type="xsd:string" />

    <xsd:element name="GenericSatelliteProperties">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="latitude" minOccurs="1"
            maxOccurs="1" />
          <xsd:element ref="longitude" minOccurs="1"
            maxOccurs="1" />
          <xsd:element ref="altitude" minOccurs="1"
            maxOccurs="1" />
          <xsd:element ref="pitch" minOccurs="1"
            maxOccurs="1" />
          <xsd:element ref="yaw" minOccurs="1"
            maxOccurs="1" />
          <xsd:element ref="roll" minOccurs="1"
            maxOccurs="1" />
          <xsd:element ref="focalLength"
            minOccurs="1" maxOccurs="1" />
          <xsd:element ref="currentView"
            minOccurs="1" maxOccurs="1" />
          <xsd:any/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

  </xsd:schema>
</types>
```

```
<message name="CreateSatelliteRequest">
  <part name="request" element="tns:createSatellite"/>
</message>

<message name="CreateSatelliteResponse">
  <part name="response" element="
tns:createSatelliteResponse"/>
</message>

<portType name="SatellitePortType"
  wsrp:ResourceProperties=
    "tns:GenericSatelliteProperties">

  <operation name="createSatellite">
    <input message="tns:CreateSatelliteRequest"
wsa:Action="http://example.com/CreateSatellite" />
    <output message="tns:CreateSatelliteResponse"
      wsa:Action=
        "http://example.com/CreateSatelliteResponse" />
  </operation>

  <operation name="getAltitude">
    <input message=
      "wsrpwsdl:GetResourcePropertyRequest"
      wsa:Action="http://docs.oasis-open.org/wsr
f/2004/06/WS-ResourceProperties/GetResourceProperty"/>
    <output message=
      "wsrpwsdl:GetResourcePropertyResponse"
      wsa:Action="http://docs.oasis-open.org/wsr
f/2004/06/WS-ResourceProperties/GetResourcePropertyResponse" />
  </operation>

  <operation name="getOrientation">
    <input message=
      "wsrpwsdl:GetMultipleResourcePropertiesRequest"
      wsa:Action="http://docs.oasis-open.org/wsr
f/2004/06/WS-ResourceProperties/GetMultipleResourceProperties"/>
    <output message=
      "wsrpwsdl:GetMultipleResourcePropertiesResponse"
      wsa:Action=
        "http://docs.oasis-open.org/wsr
f/2004/06/WS-ResourceProp
erties/GetMultipleResourcePropertiesResponse" />
  </operation>

  <operation name="checkOrientation">
    <input message=
      "wsrpwsdl:QueryResourcePropertiesRequest"
      wsa:Action="http://docs.oasis-open.org/wsr
f/2004/06/WS-ResourceProperties/QueryResourceProperties"/>
    <output message=
      "wsrpwsdl:QueryResourcePropertiesResponse"
      wsa:Action="http://docs.oasis-open.org/wsr
f/2004/06/WS-ResourceProperties/QueryResourcePropertiesResponse" />
  </operation>

  <operation name="addTarget">
    <input message=
      "wsrpwsdl:SetResourcePropertiesRequest"
      wsa:Action="http://docs.oasis-open.org/wsr
f/2004/06/WS-ResourceProperties/SetResourceProperty" />
```



```
<operation name="checkOrientation">
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="addTarget">
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="moveToTarget">
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
<operation name="removeTarget">
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
</binding>

<service name="SatelliteService">
  <port name="SatellitePort"
    binding="tns:SatelliteSoapBinding">
    <soap:address location=
      "http://example.com/satellite"/>
  </port>
</service>

</definitions>
```

Обратите также внимание на то, что мы можем добавлять произвольные элементы к `GenericSatelliteProperties` и поэтому мы можем свободно добавлять впоследствии новые свойства, такие, например, как `targetCoords`.

## 8. Итоги

В этом учебном пособии, первом в серии из четырех частей на тему WSRF, мы начали с того, что объяснили цель появления WSRF и рассказали, почему одни Web-службы недостаточны. Далее мы определили, что WS-ресурс – это объединение ресурса с состоянием, такого, как база данных или спутник, с Web-службой.

Сам ресурс описывается с помощью набора свойств, которые ассоциируются с Web-службой в WSDL-файле. Мы также коснулись основ языка

WSDL и WS-адресации, которые используются в WSRF для указания на конкретный экземпляр WS-ресурса.

Мы рассмотрели вопрос создания WS-ресурсов, выбора и настройки свойств этих ресурсов, чтобы иметь возможность манипулировать самим ресурсом.

В следующих частях этой учебной серии мы рассмотрим более развитое использование WSRF, такое, как ServiceGroups и обработку ошибок, так же как и WS-оповещение. В последней части этой серии мы все это объединим и напишем приложение, которое будет использовать классы, реализующие каждое из понятий, обсужденных в первой и второй частях этой серии.

## 8.1. Учебные ресурсы

Инфраструктура WS-Ресурс (WSRF) включает в себя спецификации, относящиеся к ряду различных тем.

Данный текст является Частью 1 в серии публикаций на тему WSRF. Обязательно прочтите пособия «Что такое WSRF», Часть 2: Как работать с *WS-ResourceLifetime* (Время жизни WS-ресурса), *WS-ServiceGroup* (Группы WS-служб) and *WS-BaseFaults* (Основные WS-ошибки), Часть 3: *Publish-subscribe with WS-Notification* (WS-оповещение), Часть 4: Использование классов Java Core WSRF.

## 8.2. WSRF и относящиеся к нему спецификации

Основным местом нахождения документации по WSRF является веб-сайт Globus Alliance (<http://www.globus.org/wsrf>), но самые последние спецификации можно найти на сайте Oasis.

Документация включает:

- Моделирование ресурсов с состоянием посредством Web-служб
- Инфраструктура WS-ресурс (<http://www.globus.org/wsrf/specs/ws-wsrf/pdf>)
- Свойства WS-ресурса (WSRF-RP)
- Время жизни WS-ресурса (WSRF-RL)
- Группы WS-ресурсов (WSRF-SG)
- Основные WS-ошибки (WSRF-BF)

## 8.3. Web-службы и относящиеся к ним спецификации

Основные рекомендации, касающиеся SOAP (<http://www.w3.org/2002/ws/#drafts>) и WSDL (<http://www.w3.org/TR/wsdl>), поддерживаются Консорциумом W3C (<http://w3.org>). IBM является одной из компаний, предложивших спецификации механизма WS-адресации. Здесь вы можете также найти полезную информацию на сайте <http://www.ibm.com/developerworks>, включая:

- Введение в Web-службы и WSDK V5.1
- Объединение грид-систем и Web-служб
- Web-службы и язык WSDL
- Использование WSDL в SOAP-приложениях
- Развертывание Web-служб на языке WSDL: Часть 1

- Какой стиль языка WSDL мне лучше всего использовать?
- Влияние SOAP-кодирования на производительность Web-служб
- Web-службы изнутри, Часть 1: Соображения, касающиеся SOAP
- Неявное влияние WS-адресации на SOAP

Вы можете также найти дополнительную информацию в разделе «Web-службы» (<http://www.ibm.com/developerworks/webservices/>).

#### **8.4. XML и относящиеся к нему спецификации**

Рекомендации, относящиеся к XML (<http://www.w3.org/TR/2004/REC-xml-20040204/>), XPath V1.0 (<http://www.w3.org/TR/xpath>), XPath V2.0 (<http://www.w3.org/XML/Query#specs>) и XML Schema (<http://www.w3.org/XML/Schema#dev>) поддерживаются Консорциумом W3C (<http://www.w3.org>). Полезную информацию можно найти также на сайте <http://www.ibm.com/developerworks>, включая:

- Обзор стандартов XML
- Введение в XML
- Проверка корректности XML
- Как начать пользоваться XPath
- Обработка XML с помощью запросов XML

Еще ряд дополнительных материалов можно найти на сайте <http://www-106.ibm.com/developerworks/xml>.

#### **8.5. Обратная связь**

Пожалуйста, дайте нам знать, оказалось ли это учебное пособие полезным для Вас, и как бы мы могли сделать его лучше. Мы были бы также рады услышать о других темах учебных пособий, которые могли бы быть освещены на сайте IBM <http://www-106.ibm.com/developerworks>.

По вопросам, касающимся содержания этого текста, обращайтесь к автору Babu Sundaram по адресу [babu@cs.uh.edu](mailto:babu@cs.uh.edu).