
el777 карма 224,1 рейтинг 0,0
447 голосов

18 ноября 2009 в 20:37

Дао Вебсервиса. (Или да хватит же изобретать велосипеды!)

Веб-разработка*

Недавно на Хабре была опубликована статья под провокационным заголовком и призывом к прекращению изобретений велосипедов в API-строении. Поскольку тема мне интересна, то я просто не мог пройти мимо.
Увы, реальность за хабракатом меня сильно разочаровала — я увидел очередной велосипед, да еще и с квадратными колесами. (Коллеги, ничего личного, только техническое обсуждение.) Правда, авторы честно сказали, что увидели на нескольких сайтах модное слово REST и решили сделать по нему. Только вот поняли они этот «РЭСТ» по-своему, примерно как Дед Щукарь читал и понимал толковый словарь.
В этом топике я призываю по-настоящему покончить с велосипедами в API сайтов. Ведь получается какой анекдот: АПИ разрабатывается для упрощения доступа к сайту и легкости подключения внешних систем, а получается такой, что с ним еще сложнее, чем без него :)

Чуть ниже под катом я подпишу смертный приговор всем велосипедам в универсальных API. Чтобы не быть голословным, я все проиллюстрирую примерами.
Но должен предупредить сразу — после прочтения статьи вы не сможете без рвотного рефлекса смотреть на очередной велосипед Васи Пупкина под гордым названием «универсальное API сайта».

В повествовании будут рассмотрены следующие вопросы:

Итак на сегодня наиболее распространенными способами доступа к API сайтов являются:
1. XML-RPC.
2. REST (с оговоркой, что это не протокол, а подход).
3. SOAP.

Базовые технологии и сравнение

XML-RPC

Думаю, не сильно ошибусь, если скажу, что XML-RPC — прародитель всей этой галиматьи, которую мы сейчас называем громкими словами «веб-сервисы» и «сайтовое api». Разрабатывался в 1998 году по заказу Микрософта. По своей сути это реинкарнация старого доброго RPC с использованием форматирования сообщений в XML. Несомненным преимуществом протокола является его простота, и как следствие, легкость реализации.
Вот пример типичного XML-RPC запроса:

```
<?xml version="1.0"?>
 <methodCall>
   <methodName>examples.getAirportCode</methodName>
   <params>
     <param>
         <value><i4>567</i4></value>
     </param>
   </params>
 </methodCall>
```

На что можно получить такой же простой ответ:

```
<?xml version="1.0"?>
 <methodResponse>
   <params>
     <param>
         <value><string>SVO</string></value>
     </param>
   </params>
 </methodResponse>
```

Примерно так же просто выглядят сообщения об ошибках. Такой ответ легко распарсить даже человеку, не говоря уже про машину. Такая простота сослужила ему двоякую службу: с одной стороны он не был принят заказчиком и не стал стандартом, а с другой — понравился толпе простых незамороченных программистов. Этим ребятам нужно было простое и надежное средство обмена информацией между системами, они не хотели заморачиваться на такую хрень как красивый УРЛ, схема документа и прочие академизмы. Первое, что они хотели получить — простую работающую систему. И до сих пор XML-RPC помогает им в этом.
Итак:
+ : простота, краткость сообщений, минимальная проверка формата данных,
— : недостаточная строгость, требуется отдельное описание сервиса.

REST

Наверное, Рой Филдинг был первым, кто сказал «хватит изобретать велосипеды» и «все уже украдено до нас» применительно к веб-сервисам. А может и не первым, в любом случае его слова прозвучали наиболее громко. В своей диссертации Architectural Styles and the Design of Network-based Software Architectures он описал базовые принципы REST (Representational State Transfer) — архитектура веб-сервисов, изменившая представления разработчиков на 180 градусов**. Он сказал, что «танцевать надо не от угла, а от печки» — иными словами — надо не процедуру вызывать и передавать ей объект, а обращаться к объекту. Не надо наворачивать велосипед на мопед. Ведь самом протоколе HTTP уже есть ряд методов работы с объектами: GET (получить), POST (отправить/создать), PUT (обновить), DELETE (удалить).
Зачем заниматься тавтологией и вызывать

```
POST /api/object.php?object_id=445&action=delete&user_id=255&auth_key=0Jf4tet5 HTTP/1.1
```

Когда можно просто:

```
DELETE /objects/445 HTTP/1.1
```

Или еще практикуется вариант, когда делается POST на сам ресурс, а delete передается отдельно параметром:

```
POST /objects/445 HTTP/1.1
```

При этом, если XML-RPC использует из HTTP-протокола только транспортную часть для передачи XML-ки запроса/ответа, то REST задействует HTTP по-полной: здесь и авторизационные заголовки, content-negotiation — предпочтения по формату, языку, кодировке и виду ответа, различные служебные заголовки, безгеморная передача бинарных данных и т.п. Ошибки хорошо описываются кодами HTTP 4xx и 5xx. Можно видеть, что REST — органичная надстройка над HTTP. Это неудивительно ведь Рой — один из разработчиков протокола HTTP. Вообще, чем больше я разбираюсь с этим протоколом тем больше мне кажется, что он опередил свое время лет на 20, и чем дальше развивается веб, тем больше возможностей мы из него используем.
Само тело сообщения может передаваться в разных форматах: классическом XML либо гиковском JSON. Вообще, REST это не протокол, это подход, и как раз здесь заключена его гибкость, он как бы говорит нам: «Ребята, берите базовые принципы, а дальше делайте как вам удобно». Близость к HTTP-протоколу упрощает и ускоряет его обработку веб-серверами.
Итак:
+ : гибкость, простота, скорость обработки (особенно важно для крупных сайтов), органичность протоколу, мультиформатность, компактность.
— : отсутствие строго контроля данных, из практических соображений приходится выходить за рамки идеальной модели.

SOAP

Вот мы и добрались до него — протокол-подарок! Но именно на нем я постиг дао всех веб-сервисов и сайтовых апи. Думаю, все в курсе, что значит аббревиатура SOAP — Simple Object Access Protocol. Именно так в представлении Микрософта должен выглядеть идеальный протокол для веб-сервисов. Давайте посмотрим на что похожи типовой запрос:

```
Copy Source | Copy HTML
```

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/
                        schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <req:echo xmlns:req="http://localhost:8080/axis2/services/MyService/">
      <req:category>classifieds</req:category>
    </req:echo>
  </soapenv:Body>
</soapenv:Envelope>
```

и ответ:

```
Copy Source | Copy HTML
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/
                        schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <wsa:ReplyTo>
      <wsa:Address>schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:Address>
    </wsa:ReplyTo>
    <wsa:From>
      <wsa:Address>localhost:8080/axis2/services/MyService</wsa:Address>
    </wsa:From>
    <wsa:MessageID>ECE5B3F187F29D28BC11433905662036</wsa:MessageID>
  </soapenv:Header>
  <soapenv:Body>
    <req:echo xmlns:req="http://localhost:8080/axis2/services/MyService/">
      <req:category>classifieds</req:category>
    </req:echo>
  </soapenv:Body>
</soapenv:Envelope>
```

— *Фак май мозг!* — воскликнете вы и будете абсолютно правы — *Это что же, ради передачи фитюльки на 10 байт мне надо всю эту лабуду писать?*
— *Да* — скажет Логика, и вы, засучив рукава пойдете учить всю эту кучу технологий.

— *Сюда еще и XML Schema приплели зачем-то? Какого хрена? А вы уверены, что эти ребята правильно понимают смысл слова «Simple»?* — такие мысли будут вас посещать, и это хорошо — вы на верном пути.

Но и это не все: чем больше копаешься в SOAP тем больше всяких разных технологий вылазит на тебя. Когда вы дойдете до WSDL (Язык описания веб-сервисов), вы поймете почему, начиная с какой-то версии, разработчики перестали воспринимать название SOAP как аббревиатуру, а начали понимать буквально — soap («мыло»). В этот момент ваши мысли будет занимать одна идея: почему во всем этом зоопарке технологий отсутствует технология горе («веревка»).

Еще через какое-то время вы воскликните: «Ну нафиг, давайте уж без меня: сами придумали — сами и возитесь. Я вам не машина мегабайты иксэмеля в мозгу парсить!».
Поздравляю: теперь вы постигли дао вебсервисов!

Да! **Дао веб-сервиса именно в этом и состоит: это язык общения машин и человеку нафиг не надо туда лезть.** Надо просто его использовать. Ведь когда вам нужна какая-то программа, вы ее запускаете, а не лезете в бинарный код, чтобы исполнять его в мозгу. Точно так же вы не отправляете HTTP-запросы руками в командной строке, а используете браузер. Так зачем здесь лезть в эту гопу голыми руками, да еще хвастаться кто залез по локоть, а кто по самое плечо? Надо просто его использовать.

Просветленные API

API сайта и веб-сервисы — это не что-то милостиво спущенное на нас с небес создателями сайта! Это банальная библиотека функций, которую мы можем встроить в свою программу. Этот вывод становится совершенно естественным, когда вы начинаете мыслить глобально за пределами своего компьютера.

Если вам нужна новая уже готовая либа в проекте, что вы делаете? Скорее всего просто скачиваете, устанавливаете и используете? Пишете в начале программы use/require/import/include/… а дальше просто вызываете функции. Почему же работа с веб-библиотекой должна быть сложнее?
Вот теперь, просветлившись, мы можем начать работать с просветленным API.

Сейчас в качестве примера мы 1 минуту сделаем работу с API Аэрофлота, будем подглядывать за табло Шереметьево без отрыва задницы от стула. Я беру свой любимый язык и на нем пишу все примеры. Уверен, в вашем языке есть аналогичные модули. Ну а если их там нет, то самое время задуматься, так ли взаимна ваша любовь ;-)

Вот тут аэрофлот подробно расписывает свое чудесное API. На первый взгляд, описание достаточно убогое — чисто перечисление методов и параметров. А как же это вызывать, как парсить, что вообще делать? А это уже не наши заботы — пусть об этом болит голова у машины — она же железная, а свою мне не хочется грузить фуфлом. Поэтому моя задача найти машинночитаемое описание сервиса — тот самый WSDL и скормить его компу.

```
Copy Source | Copy HTML
 1. <pre>
 2. from ZSI.ServiceProxy import ServiceProxy
    #  импортируем замечательную либу Zolera Soap Infrastructure
 3. api = ServiceProxy('http://webservices.aeroflot.ru/flightstatus.asmx?WSDL')
 4. # ... и все! API сайта полностью готово к работе.
 5. # наш объект api содержит методы
    AirportInfo()         Departure()
    AirportList()         FlightInfo()
    Arrival()               FlightSearch()
 9.
10. # то есть все те, что описаны в доке.
11. # Нам осталось лишь вызвать их с нужными параметрами:
12. airport_list = api.AirportList()
13. airport_list
14. {'AirportListResult': {'Airport': [{'city': 'Colombo',
15.                                    'code': 'CMB',
16.                                    'id_country': 'SRI',
17.                                    'name': 'Bandaranayake'},
18.                                   {'city': 'Belfast',
19.                                    'code': 'BFS',
20.                                    'id_country': 'UK',
21.                                    'name': 'Belfast Intl'},
22.                                    .....
23. # Обратите внимание насколько все просто. Мы в 2 команды подключились к API и легко вызываем его мет
24. # Работать мне с ним так же легко и просто, словно это обычная библиотека на моем компе.
    # Для моей программы это абсолютно прозрачно.
25. # Нас интересует табло прибытия за 15 число:
26. import datetime
27. date1 = datetime.date(2009, 11, 15)
28.
29. arrival = api.Arrival(code='SVO', date=date1, order_field='airport', order='asc')
30. arrival
31. {'ArrivalResult': {'Flight': [{'airport': 'Adler/Sochi',
32.                                'calc': (1, 1, 1,  0,  0,  0,  0,  0,  0),
33.                                'company': 'SU',
34.                                'fact': (1, 1, 1,  0,  0,  0,  0,  0,  0),
35.                                'flight_no': '874',
36.                                'flt_pk': '2009101359805123',
37.                                'is_board':  0,
38.                                'is_check':  0,
39.                                'plan': (2009, 11, 15, 10, 55,  0,  0,  0,  0),
40.                                'real': (1, 1, 1,  0,  0,  0,  0,  0,  0),
41.                                'sched': (2009, 11, 15, 10, 55,  0,  0,  0,  0),
42.                                'status': ''},
43.                                ......
44. </pre>
```

Стойте-стойте, а как же вся эта лабуда с XML, REST, передачей параметров, парсингом ответов и всеми прочими атрибутами «крутого» API?

Лучше всего на это отвечает фильм «Матрица», кадр из которого мне захотелось вынести в заголовок:

> *— Нет никакой ложки, Нео.*

Только перестав думать о ~~ложке~~ сайтовом API как о чем-то реальном, сложном, можно начать ~~гнуть ее~~ комфортно использовать.
Это и есть просветленное API — прозрачное и светлое настолько, что вы его не замечаете, когда работаете. Для вас это просто локальная библиотека. А вся остальная механика с сетевыми заморочками происходит где-то внутри и вас не напрягает.

Как отличить сайтовое API от говна.

Полагаю, внимательный читатель уже догадался?
Когда вместо простой, прозрачной и удобной работы с API сайта вам приходится морочиться с тем, как бы отправить запрос этому сайту и почему он не хочет принимать так старательно сформированный с помощью curl-а запрос, то вывод должен быть однозначным. Вам подсунули неправильный шоколад.

Выводы

В наше время наличие API для сайта претендующего на внимание программистов, всевозможные интеграции и прочее — не повод для гордости, а предмет первой необходимости. Но мало сделать API, даже если вы используете самые модные принципы — надо его сделать удобным и прозрачным. И технология передачи данных здесь имеет значение 10-й важности — это вообще не нужно прикладному программисту. Он должен просто взять и использовать вашу библиотеку.
Если вы хотите получить его внимание, чтобы он потратил свое время на интеграцию с вами — сделайте первый шаг — потратьте время на него. Дайте ему очень простую и понятную библиотеку.
Не надо кивать на то, что «мы сделали как твиттер — дали REST-подобный интерфейс». Вы забываете главное — у твиттера на каждый язык программирования по 5-10 библиотек, которые можно просто скачать и использовать не заморачиваясь на протокол rest/xmlrpc/soap.

Удачи и приятных интерфейсов!

вебсервис, webservice, api, апи, soap, python, zsi, смерть велосипедам, желчегонка, критика

+263 19369 500 el777 0,0

Комментарии (206)

jcdenton_dx 18 ноября 2009 в 20:46 # +21

Спасибо, автор

el777 18 ноября 2009 в 20:47 # ↑ +14

Спасибо благодарному читателю, осилившему пост :)
Я как опубликовал — увидел какого нереального размера оно получилось :)

atd 18 ноября 2009 в 20:59 # ↑ +9

Пост классный. А размер — самое то, кому интересно всё равно дочитает

3ds 19 ноября 2009 в 07:19 # ↑ –1

Даже не заметил как подошел к концу! Теперь я знаю Дао! Да пребудет со мной сила)

homm 18 ноября 2009 в 21:17 # ↑ 0

Уважаемый, почему статья до сих пор в личном блоге? Нужно срочно исправлять. :)

el777 18 ноября 2009 в 22:22 # ↑ +1

Перенес в веб-разработку :)

cashby 18 ноября 2009 в 23:11 # ↑ 0

Читается отлично несмотря на размер, так что благодарных читателей должно быть много.

Peregrinus 19 ноября 2009 в 02:38 # ↑ 0

Размер не имеет значения когда так обстоятельно и понятно всё разложено по полочкам. Спасибо.

НЛО прилетело и опубликовало эту надпись здесь

Bardt 19 ноября 2009 в 07:50 # 0

Дело, как говорится, не в размере, а в том, как этим размером пользуешься. Здесь автор с большой отдачей использовал весь объем статьи. На самом деле, одна из немногих длинных статей, которые я без напряга дочитал до конца.

slatvick 20 ноября 2009 в 12:16 # 0

Многим надавили на хронический «синяк» ;) Давите сильнее, чаще, безжалостнее!

homm 18 ноября 2009 в 20:55 # +7

Автор, вы мой герой дня. Думаю не не одному нужна была такая встряска.

Atv 18 ноября 2009 в 20:58 # +7

Добавлю про SOAP — одним из плюсов работы с веб-сервисами является то, что на основе WSDL (Web Services Description Language) для очень многих языков существуют генераторы так называемых «прокси классов». Это когда по WSDL генерируется код сложных объектов параметров и код для вызова методов веб-сервиса так, как будто это локальные вызовы методов веб-сервиса.

shai_hulud 18 ноября 2009 в 21:22 # 0

и для двух языков которые в основном хостят веб-сервисы (.NET, Java) есть замечательные технологии для генерации серверной части веб-сервиса. Очень удобно.

dieinzige 18 ноября 2009 в 22:18 # +9

WSDL,SOAP
Ересь, после пары 10мб схем вам действительно захочется выкинуть их — и это правильно
А грамотное api — это не то когда — Запущу это программму, она сгенерит это — я скормлю этому а потом запрошу вот это — и с помощью вот той либы я получу нужный мне ответ
и весь WS-
это решение несуществующей задачи
/
rest — лучшее решение

el777 18 ноября 2009 в 22:21 # +3

Так в том-то и дело, что не нужно этим мучать себя мегабайтами XML-я — пусть этим займутся машины.
А задача более чем жизненная — я привел пример.
Мне понадобилось ровно 3 строчки, чтобы сделать нужное.
С rest-ом так быстро и просто получится?

Siddthartha 19 ноября 2009 в 04:35 # –2

извините, я согласен с теми кто вам благодарен, хорошая статья и вы все правильно рассказываете, но потом вдруг вы демонстрируете пример и мне перестает нравится. да может быть вам нравится подключить библиотеки и вы рады что для вашего любимого языка она есть, но «чистый» (если я правильно понял смысл так как вы его описывали) rest — имхо всетаки лучше. Я даже не знал до прочтения вашей статьи что это так называется, но давно тяготею к такому подходу.
Я не хочу подключать этот soap даже если билиотека для php есть, ибо меня подташнивает от всего этого корпоративного кода который я подключаю и не дай бог туда забрести во время отладки потом.

может я один такой конечно, но соап повторит судьбу корбы — я пошлю его подальше.

SHSE 19 ноября 2009 в 13:11 # 0

Зависит от технологии. В ADO.NET Data Services (реализация REST от Microsoft) это также легко как с SOAP.

el777 19 ноября 2009 в 13:22 # 0

В каком виде вы передаете описание интерфейса в DS?

SHSE 19 ноября 2009 в 13:38 # 0

Conceptual Schema Definition Language (CSDL)

el777 19 ноября 2009 в 13:45 # 0

Я так понимаю, она не является стандартом W3C?

prn 20 ноября 2009 в 18:39 # 0

более того, ГОСТом тоже не является :)

el777 20 ноября 2009 в 19:20 # 0

Тогда зачем его использовать? :)

Если серьезно, то WSDL имеет то преимущество, что не ограничен рамками одной платформы.

sse 18 ноября 2009 в 22:22 # ↑ +5

Вы путаете. REST не лучше, REST — просто другой.
— XML-RPC — императивный вызов процедур в чистом виде
— REST — обращения к децентрализованным ресурсам (именно к источникам данных, а не к объектам/методам) в сегменте http.
— SOAP — императивный messaging и прочее, в том числе ассинхронное.

По крайней мере, пока что это так.

avorobiev 18 ноября 2009 в 22:52 # ↑ +7

REST существенно проще для понимания.

dieinzige 18 ноября 2009 в 22:55 # ↑ +1

А кроме простоты понимая + удобства использования(что очень рядом) что еще нужно от api?)

sse 18 ноября 2009 в 23:38 # ↑ +3

От сферического api в вакууме больше ничего не надо, тут вы правы

el777 18 ноября 2009 в 23:00 # ↑ +4

Мне кажется, что REST проще вначале — его проще понять, но работать потом проще с SOAP+WSDL.
Что может быть проще написать 2 строки инициализации, и дальше использовать как локальную либу?

avorobiev 19 ноября 2009 в 10:50 # ↑ 0

WSDL хорош для сложных задач — когда есть двухсторонний обмен данными, однако требует погружения в тему.
Предложение использовать WSDL вгонит в ступор разработчика средней руки :-(.
Для задачах получения данных по запросу REST — самое то: просто, дешево и работает :-).
REST позволяет:
— делать массовые решения (а значит понятные болшинству с не высокой ценой);
— эффективно кэшировать данные и выдерживать высокие нагрузки.
Если честно, то я не знаю ни одного массового использования WSDL, так, чтобы тысячи сайтов использовали такое API.
При этом все мы знаем такие примеры для REST: amazon.com, facebook, twitter. Из российских — cbr.ru

el777 19 ноября 2009 в 11:11 # ↑ 0

На западе WSDL применяется там, где финансовые транзакции транзакции: биржи, банки.
Пример масштабного проекта — биржа ставок BetFair. API реализовано на SOAP поверх HTTPS.

allter 19 ноября 2009 в 12:21 # ↑ 0

SOAP хорош, когда надо быстро организовать RPC-взаимодействие с простыми типами данных аргументов/результатов между конкретными экземплярами ПО. ЕМНИП, когда я последний раз игрался с SOAP как со схемой, даже дотнетовские компоненты довольно специфически понимали XML Namespaces.

Т.е. реально смысл фразы «реализовать SOAP-интерфейс» — это «реализовать интерфейс для RPC-вызовов из дотнета». А вот когда приходится делать усложнения, например, привязывать шифрование к конкретным запросам (ws-security, иначе серверу придётся привязываться к информации об SSL`е от веб-сервера) наступает совсем тяжко.

el777 19 ноября 2009 в 12:38 # ↑ 0

Да, ws-security — вещь в себе.
Я видел упощенное решение в таком виде: человек авторизуется логином паролем, получает некий токен (по сути — куку сессии) и дальше указывает эту куку в каждом запросе.

2king2 3 августа 2012 в 13:15 # ↑ 0

По такой схеме работал с веб-сервисом компании NAVTEQ (назывался не куку сесии, просто ticket), только им GPS координаты. Visual Studio указал url сервиса, студия сгенерировала прокси-класс и мне осталось только написать две строчки по работе с сервисом используя два основные методы GET и SET. И все никаких тебе xml схем на мегабайты я не рисовал и не парсил в голове!

НЛО прилетело и опубликовало эту надпись здесь

el777 18 ноября 2009 в 22:21 # ↑ 0

Исправил. Спасибо.

Razunter 18 ноября 2009 в 23:46 # ↑ +9

й**нулись :)

oleg_podsadny 19 ноября 2009 в 12:03 # ↑ –1

Украинизированый вариант, йо :) Слог автора тоже надо сказать крепкий, но мне нравится когда у разработчика есть, как говорят «strong view».

naishin 19 ноября 2009 в 16:09 # ↑ 0

Да, да. Еще «выйгрыш». Меня это просто в ступор вводит. На этой волне у меня смешанные чувства стало вызывать слово «Детройт» :)

CheatEx 18 ноября 2009 в 21:12 # +5

Теория хороша, конечно. Но ситуации когда либы работы с мылом не дружат и надо смотреть реальные сообщения и работать над ними напильником вполне себе регулярны. Тогда концепцией REST проникаешься сразу и нафиг.

tegger 18 ноября 2009 в 22:35 # ↑ 0

Да уж, граблей там разложено слишком дофига. Важные детали не стандартизованы. Приходится то допиливать [де]сериализаторы, то разбираться с кривыми серверами, использующими свой ip-адрес в качестве xmlns, то еще с чем-нибудь бороться.

el777 18 ноября 2009 в 22:37 # ↑ +1

Кривая реализация может встретиться везде.

tegger 18 ноября 2009 в 23:01 # ↑ +2

Да кто ж спорит. Но SOAP отлаживать особенно сложно.

el777 18 ноября 2009 в 23:37 # ↑ –2

Мне кажется, что в SOAP все будет более надежная система — потому что жестко формализованы (в XML Schema) запрос и ответ — то есть, если вам с того конца отправили неверный ответ, то вы получите ошибку, а не бредятину. А вот в REST вам вместо инфо об аэропорте могут легко прислать картинку :)

arty 18 ноября 2009 в 23:45 # ↑ –1

content-type negotiation

el777 18 ноября 2009 в 23:55 # ↑ +1

Это в норме. А если на той стороне в этот момент что-то внедряли, правили...? То вам легко могли прислать что угодно.
А WSDL позволяет задать жесткую проверку. Конечно, ее можно реализовать самостоятельно — но зачем если есть готовый инструмент?

tegger 18 ноября 2009 в 23:57 # ↑ +2

WSDL — это просто описание, оно может точно так же не соответствовать реальности, как и документация к REST-сервису. Так что картинку вместо информации об аэропорте можно неожиданно получить и в SOAP (и наверняка найдутся WS-клиенты, которые эту картинку попробуют отдать программе под видом аэропорта).

tegger 19 ноября 2009 в 04:29 # ↑ 0

Ну то есть я понимаю, что автогенерация парсеров — это удобная штука и сильный аргумент в пользу SOAP. Но проблему излишней сложности всего WS-хозяйства она не решает.

CheatEx 19 ноября 2009 в 01:03 # ↑ –4

Именно! В точку. Поэтому протоколы и строятся всеми максимально простыми. Всеми, кроме MS…

НЛО прилетело и опубликовало эту надпись здесь

CheatEx 19 ноября 2009 в 13:25 # ↑ 0

Не понял вопроса.

CheatEx 19 ноября 2009 в 13:24 # ↑ 0

Нихрена не понял. Общественность считает что сложный протокол легче в отладке или что SOAP/WSDL относятся к

простым?

vladem 19 ноября 2009 в 13:41 # ↑ 0

Ну почему же сразу про MS? А как же, например, IBM? У них тоже есть своя реализация SOAP. И проблема не в кривизне, а в несовместимости этих реализаций. В рамках одной платформы все работает хорошо. Если говорить про MS — при использовании **с обеих** сторон

vladem 19 ноября 2009 в 13:43 # ↑ 0

не дописал…
… с обеих сторон (и в вызывающей, и в вызываемой) одной платформы, работать удобнее (именно удобнее!) с SOAP, т.к. development-инструменты избавляют от рутинной работы.

dmitriko 18 ноября 2009 в 22:58 # ↑ +4

именно.
простые сисемные администраторы выбирают REST.
нам бы что б работало.

пример автора поста не заработал.

ошибка WSDLError: No address binding found in port.
вылетело со строчки 3
python2.5 ZSI-2.0_rc3

что вполне ожидаемо когда видишь слова WSDL ;)

el777 18 ноября 2009 в 23:04 # ↑ 0

Что именно не сработало? Напечатайте подробно ошибку.

dmitriko 18 ноября 2009 в 23:18 # ↑ 0

/opt/local/lib/python2.5/site-packages/ZSI-2.0_rc3-py2.5.egg/ZSI/wstools/WSDLTools.py in getAddressBinding(self)
1114 return item
1115 raise WSDLError(
-> 1116 'No address binding found in port.'
1117)
1118

glorybox 19 ноября 2009 в 00:04 # ↑ 0

боюсь что это нормально.

некоторое время назад была задача подружить perl и python подсистемы через SOAP.

Под перл нашелся неплохо поддерживаемая SOAP::Lite, а вот найти хорошую, не заброшенную либу для python оказалось проблемой. Пришлось допиливать какой-то древний SOAPy то ли SOAPpy для того чтобы он принимал и отправлял сессионную куку от SOAP сервера.

dmitriko 19 ноября 2009 в 00:29 # ↑ 0

только после совета страшего товарища

— используй только стандратные библиотеки

создавал SOAP сообщения из обычных питоновский темплейтов,
а парсил by ElementTree

так заработало нормально.

было очень смешно смотреть потом реализацию на Java.

el777 19 ноября 2009 в 00:37 # ↑ +1

Да уж…

el777 18 ноября 2009 в 23:07 # ↑ 0

Я использовал
Python 2.5.2 (r252:60911, Jan 4 2009, 21:59:32)
ZSI.version.Version (2, 1, 0)

dmitriko 18 ноября 2009 в 23:26 # ↑ 0

я ZSI 2.0 rc3 получил by easy_install

remote_data = urllib.urlopen(«https://user:password@host/rest/vm/list»)
vm_list = simplejson.loads(remote_data.read())

примерно так я получаю список виртульных машин с Enomalism2 по http

а что бы то же самое с SOAP и VMWare пришлось изрядно заморочиться.
изрядно.

el777 18 ноября 2009 в 23:40 # ↑ 0

Попробуйте обновить.
У меня убунту я ставил через aptitude.

НЛО прилетело и опубликовало эту надпись здесь

SHSE 19 ноября 2009 в 13:20 # ↑ 0

Год назад пытался найти **работающую** библиотеку на Javascript для работы с SOAP, отчаявшись решил написать свой парсер WSDL. Через пару дней понял, что сильно недооценил задачу и под натиском сроков просто перешли на REST.

SeLarin 18 ноября 2009 в 21:17 # +3

А чего в личный блог-то написал? Посту место на главной и в тематическом блоге.

sse 18 ноября 2009 в 22:19 # +3

Пост превосходен. А то заладили — SOAP то, SOAP се, даже те, кто его в глаза не видали.

docomo 18 ноября 2009 в 22:28 # +2

В качестве внешнего API, пожалуй, и стоит отдавать SOAP. Но для связки внутри сервисов одного проекта он не очень кандидат, — достаточно много ресурсов отъедает на парсинг/генерацию XML.

el777 18 ноября 2009 в 22:36 # ↑ +1

Это да, весь этот лохматый ХМЛ сьест немало ресурсов.
Тут получается, что WSDL+SOAP — это универсальное и простое решение для *внешнего* пользователя, желающего интергироваться с вашей системой. Он абстрагирован от всего, чего можно, даже от транспорта — существуют варианты SOAP-over-SMTP, SOAP-over-FTP,…
Для *внутреннего* использования он слишком избыточен и абстрактен — там лучше более конкретные протоколы, где-то даже бинарные.

НЛО прилетело и опубликовало эту надпись здесь

Zibx 18 ноября 2009 в 22:45 # +1

JSON мне милее, чем XML. Почему его даже не упомянули? (

el777 18 ноября 2009 в 22:47 # ↑ +4

JSON — это формат кодирования сообщения.
Я указал его как возможный вариант в REST.

Setti 18 ноября 2009 в 23:15 # ↑ +2

кстати json-rpc.org/

el777 18 ноября 2009 в 23:47 # ↑ +2

Да. Отличная вещь.
Вообще это говорит про гибкость SOAP — самый разный транспорт, различный формат сообщений.
Интересно, можно ли описать RESTful интерфейс в WSDL?

voidus 19 ноября 2009 в 13:49 # ↑ 0

Для RESTful интерфейсов есть свой стандарт — WADL

Zibx 18 ноября 2009 в 22:49 # ↑ 0

Я понял, что я — идиот и тут несколько другое.)

arty 18 ноября 2009 в 22:58 # +25

во-первых, хочу уточнить про PUT/POST в REST, а то в посте они перепутаны. PUT — создание, POST — обновление

во-вторых, WS-* кажется мягким и шелковистым до тех пор, пока не влезаете в legacy код, и не начинаете разгребать чужие грабли. Вот тогда вам придётся собирать эти дикие запросы руками, и вы быстро разлюбите эту семейку протоколов. И не надейтесь, что legacy код бывает только у идиотов, с которыми вам не придётся работать — я лично бился с API не какой-то шарашкиной конторы, а самого Google, у которого WSDL не соответствовал реальности, а обновить они его не могли. Очень было весело… Такая психология «инструменты всё сделают за нас» была очень популярна в первые годы XML, но довольно быстро сошла на нет. Далеко не всегда инструменты защитят вас от кромешного ужаса.

для меня самым ярким примером глупости WS-* было то, что в нём есть аналог HTTP, при том, что сам WS-* работает на HTTP. Получается такой HTTP-over-HTTP.

а, пожалуй, решающий аргумент за REST состоит в том, что он на порядки лучше интегрируется в веб, потому что он и есть веб — его очень просто, легко и эффективно кешировать, балансировать, шардить и прочая, и прочая — и всё это существующим дешёвым железом общего назначения. У нас есть перед глазами пример интернета — невероятно успешной огромной сети, и HTTP — одна из главных причин такой её масштабируемости. Глупо выбрасывать работающий и надёжно проверенный инструмент. Пользуйтесь HTTP, а не WS-*/SOAP

я не буду дальше в комментах что-то доказывать, просто для пользы хабра хочу, чтобы тут были эти слова


el777 18 ноября 2009 в 23:24 # ↑ +1

Спасибо за интересный коммент.

По методам: мне попадались разные варианты, и на, мой взгляд, по названию PUT тоже больше подходит для создания.

Я считаю, что любой инструмент надо поддерживать в порядке. Кривой WSDL — кривой интерфейс. Проблема часто решается так: делается несколько версий API, и разные версии WSDL. Это удобно тем, что вы для себя раз и навсегда можете зафиксировать интерфейс, при этом компания может выпускать новые версии.

Да, WSDL сильно абстрактен и избыточен — но вы его можете гонять даже по FTP или почте.

Угу. Обратите внимание на выбор инструментов: массовые сервисы с невысокими требованиями к надежности (типа соц. сетей) выбирают REST — с ним нагрузка меньше. А вот где идет работа с деньгами (биржи, платежные системы) предпочитают SOAP.


arty 18 ноября 2009 в 23:34 # ↑ +4

там, где идёт работа с деньгами, к вам приезжают консультанты, ведут вас играть в гольф и кататься на яхте, и между делом продают вам решения за $30 000 000 ;)


el777 18 ноября 2009 в 23:48 # ↑ 0

Тоже вариант. Кстати, быть таким консультантом тоже не так плохо ;-)
Ну а раз они продают эти решения и дают гарантию, то значит уверены? :)


Regis 19 ноября 2009 в 04:21 # ↑ +1

Они продают до первой проваленной сделки :)


unicast 18 ноября 2009 в 23:29 # ↑ +3

+1
имхо, именно поэтому фраза «мы сделали как твиттер — дали REST-подобный интерфейс» выглядит не так уж и глупо.


CheatEx 19 ноября 2009 в 01:13 # ↑ +1

> Далеко не всегда инструменты защитят вас от кромешного ужаса.
С некоторых пор у меня сложилось впечатление, что именно они ведут к этому ужасу. Плавно сложилось впечатление, что минимум проблем имеет народ строющий запросы шаблонными движками и разбирающий самописными анализаторами. А супер-(де)сериализаторами, биндилками и валидаторами и прочей халявной атоматикой вымощена дорога в ад :(
> для меня самым ярким примером глупости WS-* было то, что в нём есть аналог HTTP, при том, что сам WS-* работает на HTTP. Получается такой HTTP-over-HTTP.
Не стоит забывать что SOAP делался для широкого круга транспортных протоколов, в частности вполне проработана его реализация на основе SMTP. Отсюда некоторое перекрытие с функциями HTTP.


vitalyk 19 ноября 2009 в 03:30 # ↑ +5

Ничего как раз не перепутано. Создание — POST, обновление — PUT

По определению HTTP (пример смотреть на www.w3.org/Protocols/rfc2616/rfc2616-sec9.html):

GET, HEAD, PUT, DELETE — Идемпотентные методы. т.е. повторное их выполнение не должно визывать изменений состояния ресурса. Сколко ресурсу не говори «PUT тебя зовут вася» – один, два или десять раз – резултат один :)

POST — повторный вызов может привести к нежелательным резултатам (например создать несколко новых ресурсов).


mastak 18 ноября 2009 в 22:59 # +1

Отличный пост, спасибо!

ttim 18 ноября 2009 в 23:00 # +2

Дайте ссылку на пост про «очередной велосипед».

el777 18 ноября 2009 в 23:49 # ↑ +1

API для сайта, хватит изобретать велосипед!

maratk 18 ноября 2009 в 23:01 # +2

я, конечно, придираюсь, но «тавтология» пишется через «в»
А за пост спасибо!

el777 18 ноября 2009 в 23:10 # ↑ 0

Вы правы, спасибо.

vladsm 18 ноября 2009 в 23:10 # 0

API — «он» («интерфейс»), а не «оно»

VasilioRuzanni 18 ноября 2009 в 23:18 # ↑ 0

Ну в немецком вполне себе даже «оно» :))

vladsm 18 ноября 2009 в 23:21 # ↑ +1

В случае немецкого тогда и писать надо не API, а что-нить в духе Programmierschnittstelle (Schnittstelle zur Anwendungsprogrammierung)

VasilioRuzanni 18 ноября 2009 в 23:28 # ↑ 0

Es war der Scherz :) Тот «интерфейс», который нам нужен (Schnittstelle) — тоже мужского рода.

vladsm 18 ноября 2009 в 23:33 # ↑ +2

Да вот щаз прям :)
Die Schnittstelle — женский род.

VasilioRuzanni 18 ноября 2009 в 23:54 # ↑ 0

Ох блин, точно!) Спасибо за поправку :)

Vertex 19 ноября 2009 в 07:19 # ↑ +1

В 6 утра мало того что мозг мне сломали статьей, так еще и язык комментариями… ну вас, блин!.. :)

ostapbender 19 ноября 2009 в 11:48 # ↑ 0

Конгениально!

Frag 19 ноября 2009 в 11:22 # ↑ –2

Написано заглавными и латинскими, потому что является английской аббревиатурой, а имеет средний род, потому что по-русски читается как «апи» и имеет все признаки среднего, а не мужского рода.

allter 19 ноября 2009 в 12:36 # ↑ –1

Почему тогда не множественного числа и не глагола? :)

Frag 19 ноября 2009 в 12:49 # ↑ –1

Если бы мы использовали в качестве сказуемого, то был бы глагол, но в данной форме («апи») не имел бы признаки глагола.
А тут слово «апи» вполне себе существительное, и, так как, мы его используем в качестве описания одной сущности, то единственное число, если бы мы говорили про несколько «API», было бы множественное.

allter 19 ноября 2009 в 12:59 # ↑ –1

Несмотря на оффтопик (ухожу, ухожу), хотелось бы отметить, что «очки» тоже описывают одну сущность, а число множественное. Т.е. по форме слова нельзя однозначно судить о его параметрах в языке.

P.S. Я согласен с тем, что api — оно, также как и кофе. :)

vladsm 19 ноября 2009 в 14:08 # ↑ –1

Это какие же признаки среднего рода имеет слово API?

Frag 19 ноября 2009 в 16:21 # ↑ –1

Если мы используем какое-либо слово в русском тексте, то читать его будут по-русски.
Получается, что написано «API», а читается как «апи», что устоялось уже в русской речи в технических кругах.
Морфологические признаки прямо сейчас точно не приведу, но в этой статье немного про рода написано. Например, там есть такая фраза: «Иноязычные существительные чаще относятся к среднему роду (кафе, меню, ателье)». Так написано не потому, что слова иностранные, а потому что большое количество иностранных слов похожи на русские слова среднего рода.

vladsm 19 ноября 2009 в 16:34 # ↑ –1

Чтение API как «апи» устоялось. Вот только «АПИ», блин, это «интерфейс» — он.

Так же как «кофе» — «напиток», он.
Это только «Фурсенко» — оно.

З.Ы. Не, ну можно даже ради интереса посравнивать через тот же яндекс частоту словосочетаний «новый апи» vs «новое апи», «удобный API» vs «удобное API», «новый API» vs «новое API»

Frag 19 ноября 2009 в 16:53 # ↑ –1

Предположим, что я прав и по всем признакам это слово среднего рода, тогда вы будете сравнивать реальность и формальность.
Со временем они взаимопроникают, но в какой-то конкретный момент времени имеют более-менее чёткие границы по отношению как к конкретному слову, так и к определённым группам.
Т. е. вы не сможете ничего доказать частотой, имхо.
ЗЫ Большая часть людей говорит «звОнят» вместо «звонЯт», но на язык одномоментно это не влияет. Возможно, лет через 20 это станет нормой.
Здесь другая ситуация: апи — не словарное слово. Поэтому все склоняют как хотят, и как приживётся, так и будет.
ЗЗЫ Ещё один пример — «бьеннале» или «биеннале». Говорят, что это «выставка», поэтому «она».
На самом деле, можно «биеннале» перевести и как «фестиваль», тогда будет уже «он». Вот здесь и кроется засада. Кто-то автоматом берёт род перевода, а кто-то устанавливает род, чувствуя язык. Т. е. если взять само слово не зная его смысла, то получится «оно».
Первоначальный вариант уже начал изменяться («бьеннале» → «биеннале»), обретая более родные для русского языка черты. Со временем, как слово станет самостоятельным, скорее всего, обретёт и свой род.

Frag 19 ноября 2009 в 16:43 # ↑ 0

Одно из правил:
«К существительным среднего рода относятся: существительные с окончанием -о (-е) в именительном падеже единственного числа; десять слов на -мя: имя, время, племя, знамя, бремя, семя, стремя, темя, пламя и вымя; слово дитя.» (отсюда).
Про саму морфологию — здесь.
В общем, я не филолог и мне сложно привести точное правило.
Пока нашёл такое:
«**Несклоняемые имена существительные**, попадая в русский язык или образуясь в нем, должны приобрести родовую характеристику, которая будет проявляться только при выборе согласованных с существительным прилагательных, причастий и глаголов.
Существуют следующие закономерности выбора такими существительными родовой характеристики: род зависит либо от значения слова, либо от рода другого русского слова, которое рассматривается как синоним или как родовое наименование для данного неизменяемого слова. Для разных групп существительных ведущими являются разные критерии.
Если существительное обозначает предмет, то оно обычно приобретает характеристику среднего рода: пальто, кашне, метро. Однако женского рода авеню (так как улица), кольраби (так как это капуста), кофе — с колебанием — мужской / средний, мужской род — пенальти, евро.»
На самом деле, язык живой и что-то приживается так, что-то эдак.
Но пример с кофе, мне кажется, показательный: кофе дали несвойственный ему мужской род, а со временем слово обжилось и приобрело более точный для языка средний.
Хотя, бывает и обратное — изначально верные слова начинают использовать неправильно, не взирая при этом на все правила и признаки.
Больше по существу сказать не готов.
По ощущениям апи — оно.
У вас может быть другое мнение. Это не противозаконно и даже хорошо, что так.

vladsm 19 ноября 2009 в 17:20 # ↑ 0

> …существительные с окончанием -о (-е) в именительном падеже единственного числа; десять слов на -мя…

Это имеет какое-то отношение к слову «апи», которое, ну как мне показалось, оканчивается на "-и"?
Откуда вообще сама мысль-то про средний род появилась? Из каких соображений?

Вы же сами только что написали:

> …род зависит либо от значения слова, либо от рода другого русского слова, которое рассматривается как синоним или как родовое наименование для данного неизменяемого слова…

Значение: "**набор** готовых классов, функций и т.д. для бла-бла-бла"
Синоним: "**интерфейс**", "**программный интерфейс**" и т.п. (нерусское слово, но тем не менее уже устоявшееся в языке и с понятными «половыми признаками»)
API -> «апи» -> «интерфейс» -> мужской род.
Ну или приведите тогда уж синоним (или «родовое наименование») среднего рода для понятия «API».

> …изначально верные слова начинают использовать неправильно, не взирая при этом на все правила и признаки.

Именно это вы и демонстрируете

Frag 20 ноября 2009 в 09:42 # ↑ 0

Процитирую себя же:

> Если существительное обозначает предмет, то оно обычно приобретает характеристику среднего рода.

.
Отсюда средний род.

> род зависит либо от значения слова, либо от рода другого русского слова

Я вам рассказал как слова проникают в язык. Имхо, передача рода слову по смыслу очень странна, как я и написал.
Одно и то же слово можно по-разному перевести. Тогда какой род брать? Первого перевода или второго?
В общем, я считаю, что слово нужно сразу встраивать в язык и рассматривать (как уже обрусевшее) с точки зрения морфологических признаков и по аналогии с другими русскими словами определять род.
Т.е. именно не «интерфейс», не синоним, а просто «апи».

vladsm 20 ноября 2009 в 14:49 # ↑ 0

Предмет? Пощупал «апи» и ощутил его тяжесть…

> я считаю, что слово нужно сразу встраивать в язык и рассматривать (как уже обрусевшее) с точки зрения морфологических признаков и по аналогии с другими русскими словами определять род.

То есть вы создаёте новые, собственные законы морфологии? И на правила русского языка вам плевать, потому что *« — Ну вот чувствую я сердцем, что оно среднего рода! Объяснить не могу, сердцем чую!»*
Ещё раз вынужден процитировать вас же:

> изначально верные слова начинают использовать неправильно, не взирая при этом на все правила и признаки.

P.S. Вы ведь так и не объяснили, из каких вообще соображений возникла сама мысль про средний род…

VasilioRuzanni 18 ноября 2009 в 23:16 # +2

Отличный пост!
Суть даже не в том, что «SOAP иль не SOAP, вот в чем вопрос», а в простоте и уровне абстракции.
Вы совершенно правы насчет того, что есть вещи, по поводу которых не надо заморачиваться.

Однако я не вполне согласен с такой категоричностью. SOAP (или любой его навороченный низкоуровневый заменитель) — не панацея, все зависит от конкретной реализации. На практике ничто никогда не проходит слишком гладко. В свое время мы пришли к использованию REST именно в таком контексте — нужно было что-то простое **для восприятия**, а воспринимается лучше то, внутреннее устройство чего понятно (речь идет, конечно, о конкретном одном уровне абстракции — том, на котором мы работаем в конкретный момент; нижележащие уровни мы здесь примем как «просто работают»).

el777 19 ноября 2009 в 00:09 # ↑ +1

Я не утверждаю, что SOAP всегда будет лучше. Есть ряд задач (высокая нагрузка, некритична высокая недежность, ...) где REST окажется предпочтительнее.
REST действительно проще для понимания, и если делать «в лоб», то с ним сделаешь быстрее, чем с SOAP.
А вы разрабатывали интерфейс или подключались к нему?

А вообще мне очень хочется видеть очень простые и удобные API, чтобы работать было та же просто, как с локальными либами.

romanoza 18 ноября 2009 в 23:16 # +3

Автор, Вы — молодец. Так просто и таким легким языком описали то, о чем я боялся даже думать. Теперь не то, что бы думать, я буду использовать ЭТО.

CheatEx 19 ноября 2009 в 01:14 # ↑ +1

Храни Вас бог…

Fatalius 18 ноября 2009 в 23:18 # +7

О, «возбудил» человека на написание статьи ;)

SOAP — отлаживать сложно, не представляю как завязать OAuth с SOAP.

Велосипед — не велосипед, но думаю, что facebook и twitter придумали довольно таки не плохое API, они так и пишут, что их API использует REST-like interface.

И конечно же, спасибо за информацию!

el777 18 ноября 2009 в 23:29 # ↑ +3

Да! Именно вы меня сподвигли на написание. :)
Пока писал, еще раз для себя все освежил :)
Спасибо за повод!

Я работал с интересной системой (кстати, весьма нагруженной — они однозначный лидер в своей сфере). У них API разделено на 2 части: авторизация, где пользователь получает ключик, и состенно система, где он использует этот ключик.

akNekto 18 ноября 2009 в 23:22 # 0

Складывается ощущение что пользуется только .NET/Java и не хочет задумываться о чем-то еще. Увы, работа с SOAP, например, в C вызывает некоторые затруднения. Более того, складывается ощущение, что автор толком не понимает что такое REST. REST — это не протокол, это архитектура, следуя которой по одному и тому же адресу можно отдавать как html-страницу, так и xml, json и все что душе угодно.

el777 18 ноября 2009 в 23:33 # ↑ 0

Как раз нет — у меня пример на Python.
Где больше свободы, там сложнее писать. Здесь я сделал интерфейс в одно строку — потому что есть очень жесткое описание, по которому система сама генерит классы. Я их просто использую.

vladsm 18 ноября 2009 в 23:22 # 0

Кстати, а с чего вдруг JSON стал «гиковским»?

el777 18 ноября 2009 в 23:34 # ↑ +1

Сейчас, да, он уже стал привычным вне Javascript.
Гиковость скорее как противопоставление «академичности» и «официальности» XML.

vladsm 18 ноября 2009 в 23:38 # ↑ 0

Интересное определение… То есть всё, что не «академический XML» — гиковские штучки? :)

el777 19 ноября 2009 в 00:10 # ↑ +1

Ну это так… скорее для украшения текста, не придирайтесь :)

Sap_ru 18 ноября 2009 в 23:30 # +6

Угу, но SOAP содержит не слабую прослойку парсера и енкодера всей это красотищи. Коие, во-первых, поедают массу ресурсов (особенно если нужно наладить локальное взаимодействие в сколько-либо реальном времени, а, во-вторых, содержат два вагона багов и уязвимостей во всех известных науке реализациях. Хотя, наверное, для взаимодействия через internet вне реального времени это всё оправдано — накладные расходы оправдываются простой распространения среди пользователей и унификацией.

DmitryKoterov 19 ноября 2009 в 01:25 # ↑ +3

Про массу ресурсов — нет ли у Вас цифр? Я, например, так и не смог выделить это поедание на фоне издержек от TCP-соединений и загрузки PHP-кода (в случае использования SOAP в PHP, естественно)… у меня есть подозрение, что экономия на парсинге XML здесь — это экономия на спичках. А уж если сравнивать SOAP с Apache Thrift (опять же, нативную реализацию SoapClient в PHP с реализацией Thrift на том же самом PHP), то SOAP оказывается почему-то удобнее и быстрее с первого взгляда.

Sap_ru 19 ноября 2009 в 02:24 # ↑ +1

Нету :) Но, конечно, если говорить о связке SOAP+PHP, то SOAP определённо не будет узким местом :) А, вот если у вас хотя бы java-сервер на тысячу клиентов, то уже вопрос. Это моё ПМСМ.

kns 18 ноября 2009 в 23:41 # **–1**

Согласен с топиком целиком и каждой его буквой в отдельности, но, тем не менее, из предложенных вариантов больше всего мне нравится REST. Он заставляет прослеживать каждый этап работы с апи, что замедляет разработку, но зато, например, дает возможность контролировать все происходящие процессы.
Может, у меня такое мнение, потому что я не программист.

НЛО прилетело и опубликовало эту надпись здесь

erley 18 ноября 2009 в 23:42 # **+3**

Несколько неточностей есть. Нужно понимать разницу между форматом данных и протоколом.

XML-RPC: Пожалуй самый старый подход — данные в виде XML передаются посредством RPC (remote procedure call). Это вообще появилось с появлением XML, просто его стали использовать как формат для передаваемых параметров удалённых функций. Заметьте, что HTTP тут вообще ни при чём, это просто неточная интерпретация терминологии. И кстати, RPC как правило требует permanent connection между клиентом и сервером.

REST: Тут действительно оптимально используется протокол HTTP, со всеми глаголами. Короче, нет слов — всё очень красиво и эффективно.

SOAP: Это всего лишь стандартизированный формат передаваемых данных. Тот же XML, но с предварительно описанными в XSD типами (XSD можно вставить прямо в WSDL), сообщениями, binding-ом сообщений на методы, структурированием методов в группы и описанием собственно WebService-а. SOAP всегда ходит по какому-нибудь протоколу. У вас подразумевается HTTP, но мы однажды его пускали даже поверх Tibco, надо было :) В целом SOAP описывает то, что из себя представляет SOAP envelop и вообще, WebServices — это XML-пародия на ставшую в своё время черезчур сложной Corba. Вспомните, в корбе у нас IIOP — в вебсервисах SOAP, в корбе IDL — в вебсервисах WSDL…
В своё время я участвовал в стандартизации WS-Security в OASIS, там даже никаких токенов секурити тогда ещё не было.

А вот REST — действительно красиво и удобно.

el777 19 ноября 2009 в 00:22 # ↑ **+1**

Замечания приняты.
Действительно и SOAP и XML-RPC не привязаны к HTTP. Но в статье я делал упор на веб-приложения, поэтому и транспорт HTTP. Можно было рассмотреть остальное, но я бы загрузил читателей, и увел от основной темы. Поэтому оно осталось за скобками.

Да, SOAP и WS-*- это куча всего. Но у меня была цель показать, как это можно удобно использовать для API, при этом не пугать горой букв. Чтобы человек посмотрел как «дешево» с точки зрения разработки он получает отличный интерфейс. И тут же: десериализацию, контроль параметров и ответа. То есть получается, что используя более сложную технологию, с более высоким порогом входа, можно выиграть.

Если вы расскажете про WS-Security, то, думаю, многим будет интересно :)

erley 19 ноября 2009 в 03:45 # ↑ **+7**

Ну на статью у меня времени нет, а если на пальцах то всё довольно примитивно.
В soap-конверте есть заголовок и тело, в котором собственно передаются параметры методов и результаты. Как всегда нужно обеспечить цифровую подпись конверта и иметь возможность БЫСТРО идентифицировать клиента.
Ну подпись генерируется по криптованному уже конечному контенту и вставляется в заголовок завёрнутая в специальный xml тэг.
Остаётся поместить туда же сертификат клиента чтобы сервер мог проанализировать его в первую очередь.
Ну вот и получается, что способов шифрования существует много — RSA, Kerberos итд, для каждого способа ключи и способ дешифрации немного разные. Вот стандарт WS-Security и описывает какие тэги нужно использовать в заголовке и с какими атрибутами.
С сигнатурой та же история — способов подписи несколько, нужно указать каким алгоритмом проверять.
Поначалу мы гоняли всё внутри SSL-туннеля, но он сильно просаживал CPU при наших объёмах. Да и не только у нас, поэтому и подключились к стандартизации. Дело в том, что по заголовку можно разбрасывать конверты по разным бэкендам, различая их по ключам например. Декриптовать само тело конверта на фронтенде не требуется.
Замечу, что переговоры по стандарту проходили медленно, всем хотелось сделать что-то краткое и ёмкое. Быстро стало ясно, что не получится, хотя работу над стандартизацией до конца мы довели. Потом там ещё много «WS-» стандартов было принято, но я уже не очень за этим следил.
Дело было в 2003 году, у всех тогда был пылкий интерес к вебсервисам и интернет пестрил статьями про «поворотное событие в распределённом software». На деле оказалось, что когда всё красиво смоделируешь в wsdl, то сервера умирают под нагрузкой — столько xml нужно парсить да ещё и проверять на соответствие xsd! Короче осадок остался один — это та же корба, только переписанная под xml со всеми вытекающими последствиями.
Для веба пользуйтесь REST, люди докторские не просто так получали за это (Рой Филдинг кажется). Для некоторых не очень сложных случаев вполне можно использовать и вебсервисы, хотя новое что-то я бы под них не стал сейчас создавать, их время прошло.

Quadrix 18 ноября 2009 в 23:52 # **+3**

Вот за что люблю хабр, так это за то, что как только необходимо сделать что-то чего раньше не делал, то здесь это появляется и

рассказывается. Спасибо, автор.

 **kulakowka** 19 ноября 2009 в 00:25 # **–1**

Статья крутая) только надо бы в конце все таки вердикт для чайников написать. А то как то получилось из серии «читаешь читаешь — вроде все верно пишут — а о чем конкретно хотели сказать не понятно»…

автор хотел сказать что не важно какую технологию юзать, гланое дать разработчикам готовые библиотеки или чего?

 **Ex3NDR** 19 ноября 2009 в 00:31 # **+1**

смотрю на заголовок и жду нечто феерическое, а тут на тебе… ничегошеньки НЕ банального не нахожу…

о чем топик? просто напоминание? ну тогда да… вы справились.

 **el777** 19 ноября 2009 в 00:54 # ↑ **+1**

Статья запланирована как практическое введение в веб-сервисы. Рассмотрена философия, теория и дан работающий пример. Если вы это переросли, то хорошо, значит есть чем дополнить.

P.S. Спасибо, что не кричите БОЯЯЯЯН [|::::::::|] ;-)

 **Ex3NDR** 19 ноября 2009 в 01:06 # ↑ **0**

просто понимаете, я вообще не разрабатывал более-менее адекватных веб-сервисов, но если бы я был на месте любого проекта, которым был бы нужен достаточному количеству людей, то я бы для апи еще и библиотечки на десятке языков написал бы)

 **el777** 19 ноября 2009 в 01:12 # ↑ **0**

Вот это правльный подход!

 **stepank** 19 ноября 2009 в 00:33 # **+3**

апплодирую стоя
добавлю только пару замечаний:
1. к сожалению, сделать api на основе soap+wsdl задача весьма не тривиальная. разобраться во всех тонкостях с нуля, прямо скажем, не просто
2. единственное место, где soap+wsdl использовать не удобно (хотя, может быть, я просто что-то упустил) это javascript. для внутренних нужд оказалось проще состряпать простенький протокол на основе json

 **el777** 19 ноября 2009 в 01:21 # ↑ **+1**

Тут действительно, есть над чем попотеть.
1. С другой стороны — очень удобно вашим пользователям — сгенерил либу, закинул, работаешь.
2. Да, soap слишком абстрактен для внутреннего использования. Тут и чрезмерная формализация, жесткость, оверхед.

 **stepank** 19 ноября 2009 в 11:07 # ↑ **+1**

у нас сейчас успешно используется soap, поэтому я действительно полностью согласен — если планируется хоть сколько-нибудь сложная внешняя интеграция (скажем, более 2-3 функций), то усилия по реализации soap и wsdl полностью покрываются расслабухой на этапе непосредственно интеграции
а если вспомнить про то, что для многих языков есть не только возможность создать клиента из wsdl, но и возможность создать soap сервер и wsdl из обычных классов, функций, то получается вообще сказка :)

 **DmitryKoterov** 19 ноября 2009 в 01:20 # **+7**

Немного информации по SOAP в PHP, которая может пригодиться.

1. В PHP встроенный SoapClient и SoapServer (стандартные нативные расширения).
2. SoapClient (да и SoapServer тоже) совершенно не обязательно использовать в связке с WSDL. Можно просто объявить класс, объявить в нем методы с нужными параметрами (и возвращаемым значением произвольной структуры — хоть массива массивов массивов) и передать его в SoapServer. Точно так же, достаточно передать в SoapClient URL сервера без всякого WSDL и успешно вызвать все эти методы. Проще не придумаешь.
3. В Zend Studio for Eclipse 6.1 существует генератор WSDL-файла на основе PHP-файлов с кодом и phpdoc-комментариев (в 7.x этого средства почему-то уже нет). Работает просто великолепно! Просто пишем код, объявляем параметры функции через @param type $name и получаем готовый WSDL public-свойств класса. Причем в качестве type можно указывать как простые string, boolean и т.д., так и сложные MyClass и даже массивы объектв — MyClass[], где MyClass — это мой собственный класс (в этом случае определение этого класса попадет в WSDL-файл тоже). Я, например, никогда не рискую писать WSDL руками (именно потому, что он сложен, а утилиты — типа XMLSpy — очень кривые и сложные), но сгенерировать-то актуальную версию по классам не проблема… files.zend.com/help/Zend-Studio/generate_wsdl_files.htm и forums.zend.com/viewtopic.php?f=50&t=385 и www.zend.com/en/forums/index.php?t=msg&goto=19350&S=0c850207a026dd45faff3713958e15bf
4. SOAP в PHP — это единственный производительный (!) способ сериализовать и передать на удаленную сторону некоторую сложную структуру, состоящую из объектов, массивов, списков и т.д. так, чтобы гарантировать: там она распакуется в неизменном виде. XMLRPC-модуль в PHP на это не способен. A serialize() сериализует все подряд, включая приватные свойства, что может быть крайне неудобным. Из-за этой причины я, кстати, и выбрал протокол SOAP для своей RPC-библиотеки параллельных вызовов dklab.ru/lib/Dklab_SoapClient/ (правда, кажется, там с параллельностью что-то иногда подглючивает в

самой последней версии PHP — в libcurl что-то изменили).
5. SOAP прозрачно работает с куками и сессиями. Можно, например, сделать пару вызовов $soapClient->login('user', 'pass'); $soapClient->getFriends(), и при этом на стороне сервера getFriends() будет знать о том, что до этого выполнился login() (за счет сессий или кук).

В общем, не знаю, как в других языках, а в PHP с SOAP очень комфортно работать. Даже Zend-овский генератор WSDL мне понравился больше, чем генераторы WSDL в Java (я попробовал несколько штук).

el777 19 ноября 2009 в 01:25 # ↑ +2

Спасибо за хорошее подробное дополнение!

5. Можно подробнее? По идее в самом soap нет куков? Или ваша библиотека их обрабатывает и делает таким образом сохранение состояния подключения?

apok 19 ноября 2009 в 10:50 # ↑ 0

Я для повседневных задач предпочитаю WebserviceStudio, функциональность конечно не сравнится, но зачастую — вполне достаточно.

apok 19 ноября 2009 в 10:51 # ↑ 0

Сорри, промахнулся веткой.

DmitryKoterov 19 ноября 2009 в 23:07 # ↑ 0

Куки обрабатывает не моя библиотека, а SoapClient и SoapServer.

vadimbelyaev 19 ноября 2009 в 01:35 # +1

Хочу внести небольшое уточнение: аббревиатура SOAP не расшифровывается, начиная не с какой-то версии, а с актуальной на данный момент версии 1.2.

Плюс, хочу порекомендовать всем тем, кто имеет дела с SOAP-веб-сервисами, программу soapUI. Она позволяет делать с веб-сервисами практически всё: парсить WSDL, имитировать запросы и просматривать ответы, а также создавать мощные конфигурируемые mock-сервисы (очень полезно для разработки клиентов веб-сервисов).

Xudruk 19 ноября 2009 в 01:36 # 0

Классно написано, с юмором :) Прочел до конца, хоть и не программист.

taliban 19 ноября 2009 в 01:43 # +1

почему-то я думал что конкуренция — всегда хорошо, придумывать новые вещи тоже хорошо. а по сути ведь и соапа когда-то не было, и его придумали, тоесть изобрели велосипед… сочинять и изобретать велосипеды надо, но просто при этом надо думать что делаешь, а не банально стучаться головой о стену, надеясь на то что она разверзнется

CyrruS 19 ноября 2009 в 01:49 # +1

Ой, спасибо за пост.
Мой опыт программирования закончился в 92 году бэйсиком, но прочтя это, я начинаю задумываться о Пути Просветленных))

vitalyk 19 ноября 2009 в 03:48 # +4

мне по роду работы прихотидся время от времени интегрировать разные API.

SOAP он очень хорош для сферического проекта в вакууме :)

В реальной работе я предпочитаю REST.

Несколько причин:

* если SOAP работает то всё хорошо. А если нет – лучше застрелиться.
* REST достаточно прост концептуально что бы его можно было понять и собрать «на коленке».
* SOAP — приходится пользоваться код–генераторами. Что при изменении версии протокола визывает реальный гемор т.к. оригиналный сгенеренный код уже давно «причесали» и т.д.
* к REST API можно очень легко «достучаться» из javascript-а.
* Мне страшно даже подумать как дебаггить javascript SOAP через firebug :)
* Каждый производитель мыльной либы делает чтото слегка по своему, плюс разница версий так что часто получается несовместимость между генераторами с обоих сторон, приходится в ручную с напильником…

это толко то что сразу пришло мне в голову. причинам мыльной мозговой болезни нет конца.

apok 19 ноября 2009 в 10:57 # ↑ 0

> * если SOAP работает то всё хорошо. А если нет – лучше застрелиться.

Можно класть разные версии soap-сервисов на разные endpoints, например
версия 1.1: www.example.com/api/v1.0/service.asmx

* REST достаточно прост концептуально что бы его можно было понять и собрать «на коленке».
* SOAP — приходится пользоваться код–генераторами. Что при изменении версии протокола визывает реальный гемор т.к. оригиналный сгенеренный код уже давно «причесали» и т.д.
* к REST API можно очень легко «достучаться» из javascript-а.
* Мне страшно даже подумать как дебаггить javascript SOAP через firebug :)
* Каждый производитель мыльной либы делает чтото слегка по своему, плюс разница версий так что часто получается несовместимость между генераторами с обоих сторон, приходится в ручную с напильником…

apok 19 ноября 2009 в 11:16 # ↑ +2

Лучше застрелиться когда нечаянно нажал что-то и сообщение отправилось раньше времени...

> * REST достаточно прост концептуально что бы его можно было понять и собрать «на коленке».

REST, как писали в статье — это всего лишь подход, и в каждом конкретном случае с таким же успехом его можно сделать непонятным.

> * SOAP — приходится пользоваться код–генераторами. Что при изменении версии протокола визывает реальный гемор т.к. оригиналный сгенеренный код уже давно «причесали» и т.д.

Как вариант — можно класть разные версии soap-сервисов на разные endpoints, например:
версия сервиса 1.1: www.example.com/api/v1.1/service.asmx
версия сервиса 1.2: www.example.com/api/v1.2/service.asmx
Проблем при этом не наблюдается.

> * к REST API можно очень легко «достучаться» из javascript-а.

Почему лечге? А разве не нужно в случае REST для каждого конкретного веб-сервиса городить собственную оболочку? Насколько мне известно, существуют уже готовые библиотеки для доступа к SOAP сервисам. (Готовыми не пользовались, есть своя. Возможно lahmatiy поделится когда-нибудь).

> * Мне страшно даже подумать как дебаггить javascript SOAP через firebug :)

Тут к сожалению прокомментировать не могу, сам никогда не писал на Javascript. Но в нашей команде есть клиентские присались приложения к SOAP сервисам под Symbian (C++), J2ME, Qt, .NET Framework, .NET Compact Framework, Javascript (годжет к Vista toolbar), Javascript — (большое, самое полное по функциональности приложение), небольшие модули на PHP. И мне сложно представить писать руками обертки над вызовами не одной сотни методов, контроль «правильности» возвращаемых данных, обработку ошибок %). Как оговорился автор, зачастую машина делает за нас львиную долю работы.

> * Каждый производитель мыльной либы делает чтото слегка по своему, плюс разница версий так что часто получается несовместимость между генераторами с обоих сторон, приходится в ручную с напильником…

В большинстве реализаций REST даже не производитель, а каждый **разработчик**, делает свою реализацию.

el777 19 ноября 2009 в 11:19 # ↑ 0

Смена внешнего протокола — это очень серьезная вещь, ее нужно делать только в самых крайних случаях. Потому что тем самым вы можете сломать всех, кто к вам подключился по старому протоколу. В этом смысле лучше выпустить новую версию API и подключать новых пользователей на нее. А чтобы это не стало жутким гемором в поддержке, то API не должно кардинально меняться, должна быть преемственность.
Вобщем, думайте про апи как про *чужую* локальную либу, которую вы используете в своей программе — что будет если вдруг у нее изменятся интерфейсы? Сломает ли это вашу программу?
Но тем не менее, если надо внести изменения в интерфейс, то soap в этом поможет — т.к. вы можете сделать 2 отдельных описания, в каждое включить только те функции, которые нужно.

НЛО прилетело и опубликовало эту надпись здесь

el777 19 ноября 2009 в 11:21 # ↑ 0

А какие у вас версии установлены?
Какая версия убунты? Я проверял на Ubuntu 9.10 и на Debian — все ок.

НЛО прилетело и опубликовало эту надпись здесь

el777 19 ноября 2009 в 13:48 # ↑ 0

А как ставили убунту?
Когда я обновлялся с 9.04 у меня была масса проблем — система встала очень криво. Только после того как установил с «чистого листа» — заработала нормально.

Может в этом причина?

НЛО прилетело и опубликовало эту надпись здесь

 **ikatkov** 19 ноября 2009 в 05:37 # +2

Чувствуется, что походить по граблям стека WS* вам довелось не много. И про REST вы возможно только статьи читали.

Тем не менее — я совершенно согласен, что если задача стоит — «сделать быстро» надо делать быстро. Правильность ради красоты технического решения — декаденство в чистом виде. За него денег не взять. Если у автора быстро выходит SOAP надо пилить SOAP

 **redhummer** 19 ноября 2009 в 07:14 # 0

Автору спасибо! Редко вижу слова в поддержку SOAP :)
Правда (что тут лукавить) сервисы на SOAP удобно и легко использовать, но не разрабатывать и wsdl-описывать :-)

 **uglock** 19 ноября 2009 в 08:05 # ↑ 0

Дык пишется-то все для пользователей, в том числе и API пользователей. Вот и надо для них стараться :)

НЛО прилетело и опубликовало эту надпись здесь

НЛО прилетело и опубликовало эту надпись здесь

 **SergeyKish** 19 ноября 2009 в 14:10 # ↑ 0

Используются методы HTTP библиотеки, а она знает о кэшировании, авторизации, шифровании.
Формат запроса может быть описан с использованием WADL.

 **SergeyKish** 19 ноября 2009 в 14:05 # ↑ +1

Просто используем :)

Сервер с использованием resource_controller jamesgolick.com/2007/10/19/introducing-resource_controller-focus-on-what-makes-your-controller-special.html

```
# app/controllers/posts_controller.rb
class PostsController < ResourceController::Base
end

# config/routes.rb
map.resources :posts
```

Клиент — стандартный ActiveResource

```
# app/models/post.rb
class Post < ActiveResource::Base
self.site = «0.0.0.0:3000»
end
```

Обращаемся как к ActiveResource объекту

```
@user = User.find params[:id]
@user.update_attributes params[:post]
@user.save
```

 **atomicxp** 19 ноября 2009 в 08:15 # 0

>— Фак май мозг! — воскликнете вы и будете абсолютно правы

Именно это я про себя на английском и подумал.

 **WayBe** 19 ноября 2009 в 10:04 # 0

Спасибо. Очень здорово!

 **HounD** 19 ноября 2009 в 10:10 # +2

Желающие использовать SOAP в своих проектах могут обратить внимание на gSOAP (http://gsoap2.sourceforge.net/). Генерация WSDL, серверной и клиентской заглушки по «почти» C++ заголовочным файлам позволяют избавиться от многих головных болей. Работает эта радость практически везде, включая всяческие экзитические сочетания платформ и компилятором, короче читайте документацию. Это (чтение документации) кстати еще позволит не орать про то, что XML вообще и SOAP в частности неимоверный тормоз, а при необходимости поменять транстпорт (хоть UDP) а особо упертые смогут вообще любой серелизатор прикрутить или написать свой (да хоть ASN.1, для ценителей) правда это будет уже не совсем SOAP, но кого это волнует, если код генерируется по описанию интерфейса?

Ну и да, привет всем, кто «не осилил SOAP» (с) докладчик с HighLoad'a.

Sander80 19 ноября 2009 в 10:45 # 0

Пост осилил с интересом несмотря на объем.
По поводу сложности использования API вспомнилась мое, как возился с API Google Friend Connect.
Была у меня идея сделать так, чтобы комментарии Google Friend Connect, хранимые на серверах Google, выводились на сайт не стандартным гаджетом, а php-скриптом. Якобы это все должно было работать через OpenSocial с использованием REST; в процессе пришлось погрязнуть в какие-то ужасы, а в конце-концов разработчики (пришлось и до них добраться) признались, что через API возможно прочитать только первые 150 байт комментария.
Если честно, после этого ответа хотелось их прибить — сидишь неизвестно сколько, осваиваешь какое-то левое API, и только в последний момент выдается секрет, что толку от этого API чуть.

vshtaba 19 ноября 2009 в 11:09 # 0

много текста, а смысл один

— чем проще тем лучше

Frag 19 ноября 2009 в 11:26 # ↑ 0

Только при прибавлении простоты в использовании убавляется простота во внутренней реализации и работе в реальном времени.
Получается вы можете одной строкой вызвать метод на удалённой машине, но при этом генериться такая куча иксемеля, передаётся, и парсится, что простым этот способ не назовёшь)

el777 19 ноября 2009 в 11:33 # ↑ –1

Так вы же не сами это руками делаете?
Пусть машина об этом думает!

Frag 19 ноября 2009 в 12:52 # ↑ 0

С вами я не спорю)
Вашу точку зрения понял и согласен с ней.
Просто говорю, что «чем проще, тем лучше» не совсем соответствует данным подходам.
Проще в использовании, но сложнее в технологической реализации (хоть и автоматической).
Получается обратная зависимость.
Если основная задача в простейшем использовании — такой подход очевидно лучше.
Если задача в оптимизации быстродействия и ресурсоёмкости, то такое решение проигрывает более сложным в использовании, но простым в реализации.
Вот что я хотел сказать)

Timmmm 19 ноября 2009 в 12:17 # 0

Спасибо, было интересно почитать. Но вы упустили «архитектурные» плюсы WSDL.
Вкратце, это описание формата. По WSDL вы будете точно знать что вы должны отправить и что получите в ответ. Конечно увеличивается размер информации, но в этом случае вы получаете ясную картину как работать с этими «черными ящиками».
Балансировщик нагрузок, если у вас кластер одних и тех же сервисов. Ставите «форвард» сервер на вершине иерархии и он уже за вас сможет рулить нагрузкой на сервисы + он сможет узнать какие вообще сервисы существуют в вашей системе.
Это только малая часть «перелестей».

el777 19 ноября 2009 в 17:38 # ↑ 0

Да. WSDL — это одна из самый «вкусных» вещей в SOAP.
Благодаря жесткому формальному описанию интерфейса возможна автогенерация классов для работы — поэтому и получилось так легко подключиться и работать с Аэрофлотом.

Но, что интересно, WSDL вместе с тем одна из муторных вещей, с которыми разбирается начинающий soap-ист. Доходит до анекдота — мне на форумах попадались сообщения: «Да, soap — это хорошо, но можно как-то работать без этого wsdl? А то он меня замучал». То, есть самое интересное-то и выкидывается.
Поэтому я намаренно не стал подробно его рассматривать в статье, чтобы не пугать людей :). Чтобы изучение этого стека технологий можно было начать с простого работающего примера, а не с горы непонятных спеков. Которая часто и становится непроходимым препятствием.

mr_locke 19 ноября 2009 в 12:37 # 0

Прекрасная статья! Я и не думал, что так все просто :) Спасибо

vshtaba 19 ноября 2009 в 13:44 # –1

Масло — маслянное.

Читайте философию «форме и содержание» — это, да, вещь)

el777 19 ноября 2009 в 13:46 # ↑ +1

Поясните.
Мне казалось, что я неплохо понимаю «форму — содержание».

 **vshtaba** 19 ноября 2009 в 14:22 # ↑ 0

Проблема в базовом образовании — фундаментальном подходе.

Есть люди, которые малое значение придают первоисточникам, и, как следствие, часто «скачут по верхам».

Отсюда и «новые» велосипеды и другие вечные спутники прогресса.

 **vshtaba** 19 ноября 2009 в 14:28 # ↑ 0

Зачастую, личное мнение автора (кого и чего угодно), и борьба лидера за внедрение своих технологий — есть источник клоаки в котором крутится забавный зоопарк о котором пишет автор.

Т.е. по сути проблема в субъективном перфекционизме универсификации, с формой выражающейся во всем своем многообразии.

 **cream_brule** 19 ноября 2009 в 16:08 # 0

очень хочу все это осилить, но пока не хватает знаний :(

 **el777** 19 ноября 2009 в 17:40 # ↑ 0

Начните постепенно. Хорошо начинать с какого-нибудь работающего примера.
Не пытайтесь сразу понять весь стек этих WS-технологий — это совершенно не нужно, и очень часто сбивает людей с толку, из-за чего они бросают. Начните использовать в каком-нибудь простом проекте «для себя», а дальше последовательно разберетесь во всем.

 **cream_brule** 19 ноября 2009 в 18:08 # ↑ 0

я пытался разобраться написав приложение на FLEX, вообщем хотел повторить один сервис для нашего офиса, переписав его на flex.

 **el777** 19 ноября 2009 в 18:41 # ↑ 0

Flex-ом не занимался, точно не могу сказать.
Но вот тут есть отличное видео, как работать с веб-сервисами на Flex. Получается, все очень просто, прямо как в статье выше.

 **cream_brule** 20 ноября 2009 в 09:10 # ↑ 0

спасибо, плюсанул бы, но школота сняла карму :(

 **Shchvova** 20 ноября 2009 в 02:20 # 0

Тут критика принимается? Вы знаете что такое «дао»? Просто употребили его 9К раз в разных смыслах и странных формах. "… на нем я постиг дао всех веб-сервисов..." нормальный человек поймет это не так как вы думаете. Ну это все пустяки, если учитывать что было в тексте. А было в тексте интересно и познавательного много, правда есть вопрос
А какой крупный сервис предоставляет SOAP API? Для примера его нету ни у фейсбука (от этого существующего апи меня вообще коробит), ни у твиттера. Ну да ладно, но Google вообще закрыло SOAP доступ к поиску, как непреспективный, и открыло другой апи. Не наталкивает на размышления?

 **el777** 20 ноября 2009 в 12:22 # ↑ 0

Критика здесь принимается. Вы правы в том, что это слово имеет массу значений. Расскажите, как его поняли вы и как должен понять «нормальный человек»?
Смотря какие крупные сервисы вы имеете ввиду. Чуть выше я говорил, что REST предпочитают социалки и массовые сервисы, SOAP — там где разговор идет про деньги: биржи, ставки, казино.

 **Shchvova** 20 ноября 2009 в 18:34 # ↑ 0

Ну что вы в самом деле. Хорошого в статье намного больше чем всего остального. Спасибо.

А о дао… Дао это не такое какое хочется, это не пример, не руководство. Дао это уже существующий принцип, закономерность. Еще у каждого свой путь. Ладно, давайте закроем тему така как убеждать я никого не хочу и толку от этого не будет никакого.

 **el777** 20 ноября 2009 в 19:15 # ↑ +1

Вы удивитесь, но слово «Дао» я понимаю примерно так же как Вы описали… :)

В данном случае под Дао я имел ввиду путь как смысл существования, некое предназначение (в более высоком смысле слова), предначертание веб-сервисов. Я утверждаю, что их истинный смысл не в том, чтобы просто быть, и

показывать какие-то фитюльки, а быть удобным *интерфейсом* к сайту. То есть сами по себе они имеют очень малую ценность, но открывают доступ к функциям сайта.
Лучшее веб-апи сайта должно быть как презерватив — передавать полноту ощущений (и информации) от работы с удаленной системой, и при этом не отвлекать внимание на себя.
Да, я знаю, я выбрал довольно крепкий слог, но тем самым я хотел обратить внимание разработчиков на проектирование технических интерфейсов (веб-сервисов, апи), и донести мысль ради чего все это делается. Если кто-то написал гиганское, очень сложное апи, один учебник по работе с которым занимает 10 томов, и теперь хвалится этим, то я ему рассказываю его ошибку. Весь смысл апи — удобная работа *через* него. Если оно этого не дает, то это фуфло, а не апи.
Лучший веб-сервис — такой в котором слово «веб» почти не заметно. Вы его просто используете, оно просто работает.
На мой взгляд, вот это понимание и предназначение апи стало теряться среди разработчиков. Давайте сделаем «погиковее» — это так круто, Вася сделал так-то — значит и нам так надо. Да не это надо вам! Лучше просто сделайте удобно.

Центральная идея моего топика выражается в одном предложении:

> Дао веб-сервиса или апи — быть простым и удобным интерфейсом.
>
> (На этом оно начинается и на этом же заканчивается. Все, что сверх — то от лукавого.)

А уж как и на какой технологии это сделать — не имеет значения. Возможно, я зря прямо в этом же топике привел пример и тем самым переключил внимание разработчиков из философской плоскости в практическую. Но я не люблю пустую болтовню и декларации, поэтому я показал на практике, что все может быть гораздо проще, когда задашься такой целью. И тут же дал наводку где искать.

Спасибо за Ваше внимание.

Shchvova 20 ноября 2009 в 21:37 # ↑ +1

Да :)
И я с вами согласен. Главное что мы поняли друг дружку.

jodaka 20 ноября 2009 в 15:45 # 0

просто праздник какой-то! Давно не читал ничего на хабре с таким интересом
спасибо :)

hyborg 20 ноября 2009 в 18:06 # 0

Ну и где тут зарыт Дао? Простите, но нихрена нового.

el777 20 ноября 2009 в 18:16 # ↑ 0

Расскажите, а как Вы поняли слово Дао? И что примерно ожидали?

hyborg 20 ноября 2009 в 18:34 # ↑ 0

После введения и оглавления я ожидал магии и волшебства, которые помогут в проектировании и разработке API веб приложения, которое обезжиренное и полезно для здоровья. А а итоге, всё закончилось «ой как здорово использовать готовый soap/wsdl api».

el777 20 ноября 2009 в 19:18 # ↑ 0

В топике я дал несколько вещей:
— философию интерфейсов — «делайте проще и люди к вам потянутся» и не делайте просто так для галочки.
— пример технологии как можно создавать такие простые в использовании API.

Дальше думайте сами, как это реализовать :)

el777 21 ноября 2009 в 00:32 # ↑ 0

Про понимание Дао ответил чуть выше

eBuster 22 ноября 2009 в 22:15 # 0

А мы это проходили в универе. Даже мне, на заочном, пришлось разбираться и знать.
Ваша статья помогла понять, почему же люди воротят свои системы взаимодействия, а не используют уже давно придуманные и работающие вещи.
И ещё ещё кое что про «бери и используй» прояснилось. Этого не хватает, когда переходишь с коль-нибудь низкоуровневого программирования, на использование готовых библиотек.
Спасибо.

el777 23 ноября 2009 в 15:59 # ↑ 0

Хорошо, когда базовые вещи проходят в универе. В противном случае, человек много времени тратит на изобретательство, это полезно поначалу для развития, но когда-то надо остановиться, и заставить себя разобраться с чужими либами.
Мне такая мысль пришла в голову — очень часто можно услышать «да я лучше свое напишу, чем стану разбираться». Это не

лучшая позиция для работы в серьезном проекте — т.к. созданные велосипеды встают стеной, отгораживающей от основного направления развития отрасли. Двигаясь вбок, в какой-то момент настолько отдаляешься от отрасли, что вернуться в нее становится очень сложно. А значит не можешь использовать лучшие техники, подходы и готовые решения. Таким образом, уйдя в добровольное гетто, обрекаешь проект на голодную смерть от недостатка ресурсов.

 **TiamaT** 22 ноября 2009 в 23:27 # 0

SOAP сервисы чертовски сложно отлаживать при написании сервиса. Простых и быстрых утилит не существует.

Есть SoapUI, но она не простая и не удобная. Ничего лучше я не нашел.

Автор, ты очень крут.

 **el777** 23 ноября 2009 в 16:00 # ↑ 0

Расскажите про свой опыт отладки сервисов. Что именно вы делали? Сервис, клиент, интерфейс?

 **TiamaT** 23 ноября 2009 в 19:01 # 0

asp.net web service. Разрабатывал API и писал саму бизнесс-логику.

Хотел отлаживать самоипсным клиентом, но на его написание уходила просто туча времени. Поэтому переключился на стороннюю утилиту.

Со мной можно на ты.

 **el777** 23 ноября 2009 в 20:37 # ↑ 0

Я тоже как-то закопался в одном веб-сервисе, мне в отладке помогла очень простая утилита NetTool: запросы можно посылать как с помощье ее самой, так и подключить ее как прокси. Под виндой еще есть замечательная вещь ~~«Фигляр ПЖ»~~ Fiddler2.
Нет желания написать статью про отладку soap под asp.net?

 **TiamaT** 30 ноября 2009 в 18:53 # ↑ 0

За неттул спасибо — потыкаю палочкой.

Там писать не про что. Тема слабая и на статью не тянет. К тому же я по-русски не пишу =)

 **TiamaT** 30 ноября 2009 в 19:12 # ↑ 0

Потыкал палочкой. SOAP такой вещью отлаживать, видимо, сложно. Очень сложно и долго.

 **el777** 30 ноября 2009 в 19:44 # ↑ 0

Он показывает, что передается.
Само прложение конечно не «возьмет».

 **errno** 3 декабря 2009 в 17:24 # 0

А вписывается ли в дао веб-сервисов следующая модель:
— у поставщика сервиса (прямо скажем платежный сервис

 **errno** 3 декабря 2009 в 17:33 # ↑ 0

Вот досада, не ту кнопку нажал :(((

Вполне понятна ситуация, когда поставщик сервиса «Z» реализует свой SOAP-сервер, а клиентам раздает WSDL или готовую библиотеку, все работает, все довольны.

А насколько правильно, если наоборот, поставщик сервиса имеет у себя клиента, а всех кто хочет с ним интегрироваться, заставляет писать серверы? Причем, в качестве «спецификаци протокола» предоставляет примеры сообщений, в котрых без сомнения угадывается SOAP. Поможет ли добится от них предоставления WSDL?

Направьте на путь истинный :)) В какую сторону копать?

 **el777** 3 декабря 2009 в 19:00 # ↑ +1

Здесь получается ситуация «наборот», такое встречается пореже.
Тем не менее, думаю, тоже можно сделать по дао.
Попробуйте у них взять WSDL-ку. Предоставят или нет — зависит от них. С другой стороны, думаю, они не зря выбрали такой формат, навеняка какой-то вариант автоматизации подключения новых партнеров предполагался. Если дадут, то вы можете автоматически сгенерить классы для своего сервиса и потом к ним подключить бизнус-логику.
Если не дадут… это хуже. Реверс-инженеринг WSDL по сообщениям, может занять некоторое время. Лично я пока не встречал каких-то автоматических тулзов.

errno 3 декабря 2009 в 20:26 # ↑

0

Спасибо за ответ :)

Только зарегистрированные пользователи могут оставлять комментарии. Войдите, пожалуйста.