REST vs SOAP. Часть 1. Почувствуйте разницу

Некоторое время назад я гуглил интернет по поводу “REST vs SOAP”, прочитал пару статей и вроде бы все понял, но не почувствовал от этого никакого удовлетворения. Что-то было не так, то ли я не почувствовал основную идею, то ли просто читал, одновременно слушая новый музон и думая о новой фиче в проекте. Как появилось время, решил восполнить этот пробел, заодно написав полезную статью по этому поводу.

* Все является ресурсами с уникальным идентификатором (URL)
* Все операции клиента с сервером stateless, т.е. сервер не должен хранить вообще никакой информации о клиенте – никакой сессии
* Все запросы можно поделить на 4 типа в соответствии с [CRUD](http://en.wikipedia.org/wiki/Create,_read,_update_and_delete), причем каждому типу сопоставляется HTTP метод – Post, Get, Put и Delete
* Вся логика крутится вокруг ресурсов, а не операций

Вот с такими воспоминаниями я начал бороздить просторы интернета. Первой мыслью было, а **почему выбрано название REST?** Representational State Transfer, в переводе википедии «передача состояния представления»… Никакой картинки в голове не вырисовывается даже при четвертом вчитывании. [Здесь](http://www.xfront.com/REST-Web-Services.html) пытаются ответить на мой вопрос и даже приводят то, как Рой Филдинг (человек, сформулировавший принципы REST) сам объяснял происхождение названия. Мысль сводится к тому, что запрос ресурса с сервера переводит клиентское приложение в определенное состояние (state), а запрос следующего ресурса меняет состояние приложения (transfer). А “Representational” означает то, что ресурс возвращается не просто так, а в каком-то представлении, например в представлении для машины или в представлении для человека. Сложно, как по мне, и сбивает с толку, т.к. состояние – это как раз то, что отсутвует в отношениях клиент-сервер в архитектуре REST. Я бы назвал как-то вроде*«Стандартизированное оперирование данными»*, вот только сначала надо что-то придумать, а потом уже яркое название выбирать. А Филдинг в своей диссертации признается, что название придумано не для того, чтобы было понятно, о чем речь, а «is intended to evoke an image of how a well-designed Web application behaves». Но это ничего, не будем обижаться на уважаемого человека, мы тоже в дипломных работах часто формулировали все так, чтобы было как можно непонятнее и нельзя было придраться. [Нашлась](http://habrahabr.ru/blogs/webdev/108993/) и неплохая формулировка идеи по-русски –*«представление данных в удобном для клиента формате»*. Справедливости ради надо отметить, что пока я формулировал свои доводы о нелогичности названия, я увидел в нем некоторую логику, по крайней мере в английском варианте.  
  
Важно понимать, что **REST – это не протокол и не стандарт, а архитектурный стиль**. У этого стиля есть свои принципы. Позволю себе скопировать их с понравившегося [источника](http://www.infoq.com/articles/rest-introduction) и прокомментировать:

1. **Give every “thing” an ID.**  
   Очччень желательно.
2. **Link things together.**  
   Например, в страницу (представление) о Mercedes C218 хорошо бы добавить ссылку на страницу конкретно о двигателе данной модели, чтобы желающие могли сразу туда перейти, а не тратить время на поиск этой самой страницы.
3. **Use standard methods.**  
   Имеется в виду, экономьте свои силы и деньги заказчика, используйте стандартные методы HTTP, например GET

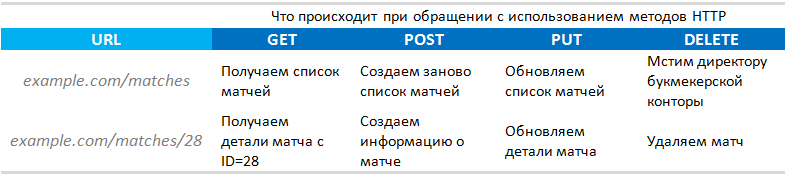
http://www.example.com/cars/00345

для получения данных вместо определения собственных методов вроде getCar?id=00345.

1. **Resources can have multiple representations.**  
   Одни и те же данные можно вернуть в XML или JSON для программной обработки или обернутыми в красивый дизайн для просмотра человеком.
2. **Communicate statelessly.**  
   Да, RESTful сервис должен быть как идеальный суд – его не должно интересовать ни прошлое подсудимого (клиента), ни будущее – он просто выносит приговор (отвечает на запрос).

Только что употребленный термин RESTful (веб-)сервис всего лишь означает сервис, реализованный с использованием принципов REST. **Так что же нам дает следование этим самым принципам REST?** Для начала я бы назвал простоту основным преимуществом архитектуры REST. Простоту идеи, простоту разработки и добавления функциональности к RESTful приложениям. Идея настолько проста и универсальна, что ее даже сложно сначала уловить. Мы не добавляем никакого нового слоя в наш и без того многослойный программерский пирог, а просто используем уже давно признанные стандарты. Поэтому чтобы ответить на вопрос о преимуществах и недостатках и чтобы анализ имел больше смысла, предлагаю перейти к сравнению подходов SOAP и REST.  
**SOAP – это целое семейство протоколов и стандартов**, откуда напрямую вытекает, что это более тяжеловесный и сложный вариант с точки зрения машинной обработки. Поэтому REST работает быстрее.

1. SOAP используют HTTP как транспортный протокол, в то время как REST базируется на нем. Это означает, что все существующие наработки на базе протокола HTTP, такие как кеширование на уровне сервера, масштабирование, продолжают так же работать в REST архитектуре, а для SOAP необходимо искать другие средства. Взамен этого SOAP сервисы получают такое мифическое свойство, как возможность работать с любым протоколом транспортного уровня вместо HTTP, однако практической пользы от него зачастую не больше, чем сотрудникам [Челябинского трубопрокатного завода](http://ru.wikipedia.org/wiki/%D0%A7%D0%B5%D0%BB%D1%8F%D0%B1%D0%B8%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D1%82%D1%80%D1%83%D0%B1%D0%BE%D0%BF%D1%80%D0%BE%D0%BA%D0%B0%D1%82%D0%BD%D1%8B%D0%B9_%D0%B7%D0%B0%D0%B2%D0%BE%D0%B4) от большого количесва статей в википедиях на [мертвых языках](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%80%D1%82%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA).
2. Есть [мнение](http://ajaxonomy.com/2008/xml/web-services-part-1-soap-vs-rest), что разработка RESTful сервисов намного проще. Наверное, это правда, если использовать Notepad в качестве основной среды разработки, но вот с использованием наших чудесных средств разработки, я позволю себе усомниться в верности этого утверждения.
3. В первом гугловском [результате](http://spf13.com/post/soap-vs-rest) по запросу «REST vs SOAP» акцентируется внимание на том, что ответ REST может быть представлен в различных форматах, а SOAP привязан к XML. Это действительно важный фактор, достаточно представить себе вызов сервиса из javascript, ответ на который мы определенно хотим получать в JSON.
4. **«REST vs SOAP» можно перефразировать в «Простота vs Стандарты»**, что проявляется в том, что для SOAP мы имеем протокол WSDL для исчерпывающего описания веб-сервиса, который с использованием все тех же чудесных средств разработки прото-таки волшебным образом делает почти всю работу за нас. Со стороны REST мы имеем загадочный и неиспользуемый протокол WADL, который, в принципе, и не нужен – он мешает простоте.
5. Второй аспект предыдущего пункта – обработка ошибок. В SOAP она полностью стандартизована, а REST может использовать давно известные коды ошибок HTTP (если здесь Вас посетила мысль, что это же очевидно и зачем я это пишу, то значит Вы внимательно читаете статью).
6. То, с чего можно было бы начать, но я припас напоследок. Это одна из ключевых мыслей. **SOAP работает с операциями, а REST – с ресурсами**. Этот факт в совокупности с отсутствием клиентского состояния у RESTful сервисов приводит нас к тому, что такие вещи как транзакции или другая сложная логика должна реализовываться «SOAP-но».

Приведу пару примеров на понимание разницы между подходами. Букмекерская контора заказала сервис для работы с футбольной статистикой. Пользовательский функционал – получить список матчей, получить детали о матче. Для редакторов – редактировать (Create, Edit, Delete) список матчей, редактировать детали матча. Для такой задачи однозначно надо выбирать подход REST и получать бенефиты от его простоты и естественности во взаимодействии с HTTP. Не нужны нам здесь SOAP-конверты, SOAP-главпочтамты и SOAP-авиапочта, которая может использовать любую марку самолета. Нам всего лишь надо реализовать следующее:  
  
  
  
Все очень просто! Теперь пример посложнее. Та же букмекерская контора захотела API для ставок на live матчи. Эта процедура включает в себя многочисленные проверки, например, продолжает ли ставка быть актуальной, не изменился ли коэффициент, не превышена ли максимальная сумма ставки для маркета. После этого происходит денежная транзакция, результаты которой записываются в основную и в резервные базы данных. Лишь после этого клиенту приходит ответ об успешности операции. Здесь явно прослеживается ориентация на операции, имеются повышенные требования к безопасности и устойчивости приложения, поэтому целесообразно использовать SOAP.   
**И еще пару задач для того, чтобы почувствовать тему:**

* Футбольный клуб заказывает CMS для подробных сведений об игроках команды-неприятеля. Нужен функционал добавления характеристик игрока простыми пользователями прямо во время матча с последующей интеграцией с табло стадиона, на котором необходимо в реальном времени отображать комментарии.
* Мексиканский наркобарон Педро Гонсалес заказывает API для учета продаж героина в Юго-Западных штатах США. Он особо просит мобильное приложение под эту задачу, т.к. его бизнес часто проходит на открытом воздухе, где нету других вариантов выхода в сеть.
* Анонимный миллиардер очень хочет такую программу, которая бы ему показывала всех его любовниц в городе, в котором он сейчас находится и то, какой текущий статус отношений. Он хочет интегрировать эту программу с уже существующим его личным десктопным приложением для подбора мест для отдыха, он очень хочет большую красную надпись о возможных неприятностях в окошке, где предлагаются варианты авиаперелета.

**Какие подходы Вы бы использовали в данных задачах?**Хотел я еще написать про то, что это все дает .NET разработчику и как это использовать в своих целях, однако вижу, что индекс нудности статьи приближается к критическому, поэтому буду закругляться. С целью понижения все того же показателя я намеренно избегал аспектов безопасности и, например, ответа на вопрос ”А как вообще возможна аутентификация в архитектуре REST, если читателю на протяжении всей этой статьи внушалось, что RESTful сервис должен быть stateless?”.  
  
А выводы статьи будут следующими:

1. **Филдинг со своими принципами REST ничего не изобрел**, а просто собрал в одну диссертацию то, что уже существовало в каком-то виде и изложил то, как можно получать максимальную выгоду из уже сформировавшейся архитектуры сети.
2. **SOAP и REST – не конкуренты**. Они представляют разные весовые категории и вряд ли найдется задача, для которой будет сложно сказать, какой подход рациональнее использовать – SOAP или REST. Поэтому «религиозные» убеждения в вопросах выбора архитектуры для веб-сервиса вряд ли будут полезны. Для тех, кто не знает, с чего начать анализ задачи, могу порекомендовать [эту](http://www.slideshare.net/cesare.pautasso/rest-vs-soap-making-the-right-architectural-decision-1st-international-soa-symposium-amsterdam-october-2008-presentation) презентацию. У них чаще побеждает REST.