13 ноября 2009 в 15:26

API для сайта, хватит изобретать велосипед!

Блог компании Badabu Media

Поступила задача – создать API для сайта scribbler.ru, которое позволило бы сторонним разработчикам работать с ресурсами сайта, писать приложения как внутри, в виде swf файла, так и находящиеся вне сайта, допустим десктопное приложение, которое может получать/отправлять почту пользователям сайта.

Всем известно, что популярные российские проекты (vkontakte.ru, mail.ru и какие-либо другие) имеют свой API. Для примера я начал было их осваивать и смотреть, как же они реализованы, и знаете что, каждый сайт пишет API так, как ему вздумается (как разработчики считают правильным), т.е. API mail.ru и vkontakte.ru сильно разнятся в своих архитектурах, грубо говоря они не схожи, что, я думаю, усложняет жизнь разработчикам, сперва пишем свои классы, работающие с API, под vkontake, потом пишем свои классы под mail.ru и т.д.

Возник вопрос: «А как же решили эту задачу «забугорные» сайты?»

Идем на twitter.com, заходим в раздел «API», смотрим, читаем и видим «The Twitter API attempts to conform to the design principles of Representational State Transfer (REST)».

Идем на facebook.com, заходим в раздел «Разработчики->documentation» и снова видим «The API uses a REST-like interface».

Да блин! Что же такое этот REST?

И вот оно! REST описывает архитектуру парадигмы для web—приложений, которые запрашивают и манипулируют web—ресурсами, используя стандартные HTTP методы GET, POST, PUT и DELETE.

Все мы знаем, что такое SOAP сервис, так REST это нечто похожее, только проще ;).

Благодаря REST мы можем обращаться к методам объектов нашего сайта, передавая параметры методами GET и POST и получая в ответ от сервера XML. Т.е., допустим, нам нужно получить информацию о пользователе, тогда мы идем по адресу .../api/user.getInfo?userId=10, в ответ нам придет XML с информацией о нем, в этом адресе наглядно видно, что мы обращаемся к объекту user и выполняем его метод getInfo, входным параметром будет userId=10, в PHP это выглядело бы так $user->getInfo(10).

Просто? Удобно? Еще как!

Ладно, хватит уже глагольствовать, пора переходить к принципу реализации, представляя, что сайт реализован на MVC архитектуре.
У API должны быть свои объекты, не стоит вызывать методы объектов, которые используются внутри сайта, в целях безопасности, думаю, это все понимают.

Итак, представим себе, что объекты API хранятся в своей папочке внутри проекта, и какой-либо разработчик обращается по адресу .../api/user.getInfo, тут врубается apiController, начинает проверять, существует ли объект user с методом getInfo, если нет, то возвращает XML со статусом ошибки, если да, то создает этот объект, вызывает нужный метод, конвертит возвращенные данные в XML и выводит в view.

Пошагово все это будет выглядеть так:

Таким образом, мы получаем API по архитектуре похожее на API twitter и facebook, которое структурировано, легко, а главное удобно разрабатывать и позволяет разрабочтикам сторонних приложений не ломать голову над курением чтением документации, и легко портировать приложения под ваши проекты.

В следующий раз продолжим беседу об OAuth авторизации, которая позволяет не вводить логин и пароль пользователя, для использования API.

Итак, теперь, сайт scribbler.ru имеет свой API, и если есть желающие опробовать его в действии, то «Welcome!»,

объективную критику и ценные пожелания расцениваем на отличненько!

Документацию по API можно прочитать здесь scribbler.ru/developers.

P.S. Старался писать как можно понятнее, история про разработку придумана и основана на реальных событиях ;)

api, scribbler.ru, rest

+38 18327 86 Fatalius 0,0

Комментарии (58)

el777 13 ноября 2009 в 15:47 # +2

Пожелание №1: Сделайте демо-доступ к API — чтобы какие-то функции можно быть запускать без авторизации. Это очень полезно для отладки, когда только начианаешь разрабатывать приложение. Если все данные нельзя отдавать без авторизации, то сделайте простенькую функцию, которая будет отдавать точное время — оно-то уж точно не секрет и меняется при каждом вызове :)

Fatalius 13 ноября 2009 в 16:30 # ↑ 0

Делаю, возможно скоро будет, ибо сейчас для разработки внутренних приложений, нужно «Создать» приложение, получить ключ и т.д, что не очень удобно.

НЛО прилетело и опубликовало эту надпись здесь

nevi 13 ноября 2009 в 15:50 # ↑ 0

Я думаю, что смысл в том, что в некоторый момент помышленными стандартами дефакто становятся решения от «монстров индустрии». В данном случае это Facebook. Может, и Вконтакте и мылу.ру придется присмотреться к REST. Это как я понял позицию автора…

НЛО прилетело и опубликовало эту надпись здесь

Unknown007 13 ноября 2009 в 22:28 # ↑ +1

Рискую быть заминусованным, но все же спрошу: А разве в том же вконтакте не REST? Ведь там так же через GET/POST посылается запрос к апи, и также ответ получается в виде XML или JSON.

Единственное, входные данные (название метода и параметры) также упаковываются в XML, как и выходные… — это единственное, что отличает данный подход от REST?

НЛО прилетело и опубликовало эту надпись здесь

ashofthedream 13 ноября 2009 в 23:08 # ↑ 0

Вы наверное плохо знаете, что представляет из себя этот апи, а я вот реализовывал его для нашего проекта, как zend_service, и поэтому могу сказать такие вот вещи — это не REST иначе вместо GET или POST (как собственно вам удобно — они равнозначны) запроса по адресу api.vkontakte.ru/api.php (уже глядя на эту строчку можно понять что это явно не наша любимая архитектура) с кучей параметров, для, к примеру взятия баланса приложения мы бы пользовались чем-то подобным api.vkontakte.ru/applications/{идентификатор приложения}…

хотим снять бабло с пользователя… и начинаем понимать что rest тут не очень уместен ;)

fornex 13 ноября 2009 в 15:49 # +4

буду занудой:

scribbler.ru/developers/details/user:

> user.denieFriendShip — Запретить дружбу.

user.denyFriendShip было бы правильней. Не суть конечно, просто режет глаз.

mdevils 13 ноября 2009 в 16:27 # ↑ +3

Если занудствовать, то user.denyFriendship еще правильнее :-)

Fatalius 13 ноября 2009 в 16:34 # ↑ +2

буду занудой и исправлю ;)

НЛО прилетело и опубликовало эту надпись здесь

+4

el777 13 ноября 2009 в 15:58 #

Вы меня извините, но в этом потоке сознания сложно разобраться.
REST — это не протокол, а подход. На его основе сделано почти все. Но это слишком общая вещь. Я бы рекомендовал использовать SOAP. Вот как разобраться в вашем примере? Он какой-то слишком наколечноый? Где описание XML Schema или хотя бы DTD? Как с этим работаь? Как вам отправлять запрос, как его парсить/понимать?

НЛО прилетело и опубликовало эту надпись здесь

el777 13 ноября 2009 в 16:12 # ↑ 0

Можно.
Мое лично мнение, что XMLRPC лучше для быстрого старта, он менее формализован, поэтому хорош, когда надо что-то быстро передать.
Для серьезной работы лучше использовать SOAP — он сложнее, формализованнее, строже, и с ним сложнее разобраться. Но потом это все окупатеся.

glorybox 14 ноября 2009 в 01:51 # ↑ 0

каким образом окупается SOAP?

Fatalius 13 ноября 2009 в 16:31 # ↑ 0

Для работы с SOAP нужно иметь свои классы на стороне клиента, что не очень приятно и удобно.

el777 14 ноября 2009 в 10:28 # ↑ 0

А чем это плохо?
Довольно приятно и удобно. Сейчас напишу подробнее.

НЛО прилетело и опубликовало эту надпись здесь

Fatalius 13 ноября 2009 в 16:27 # ↑ +1

Здесь описан принцип работы, REST не протокол, а идеология я бы сказал.

НЛО прилетело и опубликовало эту надпись здесь

el777 13 ноября 2009 в 16:44 # ↑ +1

Верно. Но если говорить про API — то оно не может быть абстрактным в отличие от идеологии.
Но на одной идеологии работать не будешь. Работать может только ее реализация.
АПИ — это конкретная реализация — в функцию getUserInfo отправил параметр userId с номерjv пользователя, в ответ получил объект типа UserInfo с полем regDate в формате RFC822 с датой регистрации этого пользователя.

Но все равно мне нравится такой вариант использования реста.

arty 13 ноября 2009 в 16:05 # +14

вообще-то ещё один из хороших принципов REST состоит в том, чтобы не дублировать в запросе методы. Когда вы хотите что-то получить, вы используете HTTP-метод GET, и нет смысла писать в урле /user/getInfo, достаточно просто /user/info. Когда вы хотите что-то добавить, вы используете HTTP-метод PUT на урл /comments, удалить — метод DELETE на тот же урл. Большинство глаголов в урлах становятся не нужны.

короче:
GET /user/info — прочитать информацию
POST /user/info — обновить информацию

GET /item/comments — получить комментарии
PUT /item/comments — добавить комментарий
DELETE /item/comments — удалить комментарий

и так далее

Fatalius 13 ноября 2009 в 16:28 # ↑ –3

Реализовать можно по разному, просто я посчитал, что так будет нагляднее, что типо синтаксиса обращения к методу объекта.

glorybox 13 ноября 2009 в 20:06 # ↑ +3

почитайте книжку RESTful web applications. там как раз описано какая разница между RPC-style API и REST.

И она как раз и заключается в переходе от вызова процедур к управлению ресурсами:
были вызовы getUser(), createUser(),…
стало

применение HTTP глаголов к ресурсу user. Стандартный CRUD при помощи HTTP методов.
В любой статье о REST об этом говорится
Create — POST
Read — GET
Update — PUT
Delete — DELETE

LighteR 13 ноября 2009 в 16:12 # +17

Похоже автор просто не понял что такое REST

sidristij 13 ноября 2009 в 16:29 # 0

Хорошо еще опираться на такие статистики:
www.programmableweb.com/apis

vasfed 13 ноября 2009 в 16:37 # +2

REST-подход вместе с MVC вообще штука хорошая, а если он был применен в движке сайта — то api на это дело тоже удобно навесить restful, тупо добавив распознавание типа входящего запроса — вся логика остается той же.
Это дело очень принято в рельсах (ruby on rails) и всем на них основанном и выглядит примерно так:

```
def update
  @resource = Resource.find(params[:id])
  @resource.update_attributes!(params)
  respond_to {|wants|
    wants.html { }
    wants.json { render :json => @resource }
    wants.xml { render :xml => @resource.to_xml }
  }
end
```

Fatalius 13 ноября 2009 в 16:42 # ↑ 0

Также это дело присутствует и в Zend framework, а рельсы определенно — всегда радуют.

skorney 13 ноября 2009 в 16:57 # 0

Возможно для js разработчиков было бы неплохо отдавать feed в формате json, а для php (php4) хорошо было бы в форме сериализированного массива. В json форме многие отдают.

Fatalius 13 ноября 2009 в 17:05 # ↑ 0

Верно подмечено и это стоит в планах, xml был выбран в качестве основного формата отдаваемых данных, т.к. его можно везде распарсить, а в будущем желаемый формат можно будет указывать, как входной параметр, в данный момент API, так сказать в стадии тестирования.

david_mz 13 ноября 2009 в 17:04 # 0

Вы, вероятно, в описании API западных сайтов дочитали только до слова REST и тут же кинулись его имплементировать.

Советую прочитать чуть дальше и дойти хотя бы до OAuth — чтобы пользователю не приходилось сообщать свой пароль любой левой тулзовине и чтобы оные пароли не гнались незащищёнными по basic authorization.

Fatalius 13 ноября 2009 в 17:05 # ↑ 0

Читайте статью до конца, пожалуйста.

В следующий раз продолжим беседу об OAuth авторизации, которая позволяет не вводить логин и пароль пользователя, для использования API.

david_mz 13 ноября 2009 в 17:10 # ↑ 0

Я зашёл на scribbler.ru/developers/auth и увидел basic auth без вариантов. Беседа — это хорошо, но речь-то о реализации.

Fatalius 13 ноября 2009 в 17:13 # ↑ 0

Да, правильно, потому, что еще не дошел до реализации OAuth.

ashofthedream 13 ноября 2009 в 18:46 # +7

И поздравляю вас, вы снова изобрели велосипед. Видимо потому, что так и не поняли что такое rest, и как вобще с помощью этой архитекруты что либо делать.

Какая привязка к объекту то? Забудьте об объектах, в rest существуют так называемые коллекции (collections) и собственно члены этих коллекций (member) + банальный HTTP-шный CRUD (собстна те самые PUT, GET, POST и DELETE) для работы с ними.

Если вы до сих пор не видете разницу между RESTful API и вашим, объясню на примере (/api/user.getInfo?userId=10);

GET /users/ — получаем всех юзеров
GET /users/10 — получаем пользователя с URI (илиже индентификатором) — 10

ну и:
DELETE /users/ — убиваем всех пользователей

stoune 13 ноября 2009 в 20:55 # 0

Всё это хорошо, но как я понял из рассылки вконтакт специально сделал свой АПИ таким и всячески приветствует отказывается от других сервисов.
Venor lock-in за исключением возможно Яндекса в рунете процветает.

enej 13 ноября 2009 в 21:07 # +1

Ох уж эти PHP программисты, вечно все исковеркают.
Автор, пожалуйста, почитайте внимательнее про REST или хотя бы комментарии к этой статьи.

Fatalius 13 ноября 2009 в 22:04 # ↑ –1

Ок, надо написать API использует принципы идиологии REST? Чтобы всем было хорошо ;)

НЛО прилетело и опубликовало эту надпись здесь

Fatalius 13 ноября 2009 в 22:19 # ↑ +2

Тогда twitter тоже противоречит, и facebook с ним на пару?

twitter:
«The Twitter API attempts to conform to the design principles of Representational State Transfer (REST)»

facebook:
«The API uses a REST-like interface»

wiki.developers.facebook.com/index.php/API

НЛО прилетело и опубликовало эту надпись здесь

ashofthedream 13 ноября 2009 в 23:11 # ↑ 0

Мне кажется что REST-like — это просто HTTP + URL + отсутсвие состояния. Собственно, можно сказать, что у нас большая часть всего в сети так или иначе REST-like

НЛО прилетело и опубликовало эту надпись здесь

Fatalius 13 ноября 2009 в 23:33 # ↑ 0

Проще говоря, каждый называет REST'ом, то, что хочет назвать REST'ом.
Спасибо, вносите ясности в дискуссию.

НЛО прилетело и опубликовало эту надпись здесь

Fatalius 13 ноября 2009 в 23:42 # ↑ 0

Прочитал, понял, доходчиво. плюс.

НЛО прилетело и опубликовало эту надпись здесь

mkevac 14 ноября 2009 в 13:39 # 0

> .../api/user.getInfo?userId=10

Как-то это совсем не похоже на REST. Правильно было бы /api/users/10.xml

el777 14 ноября 2009 в 13:58 # ↑ 0

А зачем .xml?
/api/users/10 выглядит красивее и «рестовее».

mkevac 14 ноября 2009 в 14:06 # ↑ +1

У пользователя может быть желание получить информацию в JSON, XML, CSV, еще каком-то из форматов. Он может указать желаемый формат заголовками, конечно, но неплохо бы иметь возможность выбора и в URL-е.

el777 14 ноября 2009 в 20:37 # 0

Рест вроде как предполагает Content-Negotiation.
А все равно /api/users/10/xml выглядит рестовее :)
По URI идее 10.xml, 10.json, 10.html обозначают разные сущности с разными именами, а 10/xml, 10/html и 10/json — разные подсущности (или же интерфейсы) одной сущности. Их действительно удобно использовать для явного обращения к конкретному интерфейсу объекта, в то время как просто 10 — общий вид на основе автоматических правил выбора.

mkevac 14 ноября 2009 в 22:28 # 0

Боюсь что не рестовее. Рестовее было бы /api/users/10/presentations/xml, если уж думать так, как вы думаете. Но это тоже не совсем правильно, ибо формат представления не является каким-либо ресурсом. Его нельзя удалить, добавить, изменить извне. Он есть в самой системе. Система может уметь или не уметь представлять объект в каком-либо формате. И именно расширение (.xml, .json) является самым правильным вариантом.

ITdirector 17 ноября 2009 в 00:59 # 0

А как scribbler привлекает и стимулирует сторонних разработчиков?

Fatalius 17 ноября 2009 в 09:52 # 0

В скором будущем будут задействованы внутренняя валюта — кредиты, а также было бы интересным узнать, что нужно разработчикам, для их «стимуляции»?

ITdirector 17 ноября 2009 в 11:15 # +2

Ну разработчики должны быть заинтересованы в том, чтобы писать приложения для scribbler.

У Вас же api отличается от вконтактовского, а Дуров запретил размещать приложения вконтакте в других социальных сетях, аргументировав это желанием заставить всех использовать единое api.
Это значит, что авторы приложений вконтакте просто так не пойдут на scribbler.

Вконтакте имеет следующие преимущества:
1. огромная аудитория
2. готовая система монетизации
3. удобные возможности раскрутки
4. конкурсы для разработчиков
Доход от приложения вконтакте может приносить огромные деньги.

Scribbler должен предложить какую-то альтернативу — почему разработчики будут писать для scribblera вместо вконтакта?

Fatalius 17 ноября 2009 в 11:19 # 0

Задание для маркетологов, пойду нагружу и попозже отвечу.

el777 18 ноября 2009 в 23:50 # 0

Мой взгляд на API

Только зарегистрированные пользователи могут оставлять комментарии. Войдите, пожалуйста.