Архитектура REST

Диков Д.А ВМИ-436

Глоссарий

- URL
- URI
- HTTP
- HTTP-методы
- Ресурс

Что такое **REST**?

- REST – (от англ. REpresentational State Transfer) - передача репрезентативных состояний
- REST - стиль архитектуры ПО
- Область применения – распределенные системы
- Автор термина – Рой Филдинг (Roy Fielding)
- Термин введен в 2002 г.

Зачем нужен **REST**?

- Производительность
- Масштабируемость
- Обобщенность
- Простота
- Изменяемость

REST на примере

- Задача:

  Авиакомпании необходим сервис бронирования билетов. Клиент может забронировать билет 3 типов: Эконом, Стандарт, Бизнес. В соответствии со статусом клиентов выстраивается их приоритет при обслуживании.

REST на примере

- Вариант 1 (Не “REST”)

REST на примере

- Вариант 2 (“REST”)

REST на примере

- Вариант 1 (Не REST)

REST на примере

Минусы?

REST на примере

- Веб-сервис – “узкое место” системы. Требует балансировки
- Нарушает принцип “все ресурсы в Сети должны быть уникально идентифицированы URI”
- Необходимо прописывать правила разграничения приоритетности
- Основывается на “экономии” URI

REST на примере

- Вариант 2 (REST)

REST на примере

Плюсы?

REST на примере

- Принцип действия сервиса понятен – адресная строка говорит сама за себя (принцип наименьшего удивления)
- Отсутствует “узкое место”
- Приоритеты легко расставляются – для разных классов – разные сервера с разной производительностью
- Обнаружение URI поисковыми системами
- URI однозначно идентифицирует ресурс

В чем суть **REST**?

- Передача представлений о состоянии ресурсов:

  Запрос может задать некоторому ресурсу представление о состоянии

  В ответ на запрос ресурс может возвратить представление о своем состоянии

В чем суть **REST**?

- Краткие однозначные запросы
  - URI как идентификатор ресурсов
  - URI имеет строгий формат
- Использование HTTP-методов в качестве идентификаторов CRUD
  - Create ⇔ PUT
  - Read ⇔ GET
  - Update ⇔ POST
  - Delete ⇔ DELETE

В чем суть **REST**?

- Запросы не аккумулируют никакой информации о состоянии сервера

- => Запросы кэшируются

В чем суть **REST**?

- REST основывается на протоколе HTTP
- Протокол HTTP:
  - Клиент-серверный протокол приложений, не хранящих информацию о состоянии
  - Унифицированный формат запросов для основных действий (HTTP-методы GET, POST, PUT, DELETE)
  - Запросы адресуются определенному URI
  - HTTP-запрос – конверт, содержащий заголовок, за которым следует описание ресурса (его состояния)

В чем суть **REST**?

- Соответствие является рекомендацией
  - можно определить свои действия для методов (нежелательно)
- “Смутное предназначение” POST
  - *The POST operation is very generic and no specific meaning can be attached to it”*
  - POST можно использовать по своему усмотрению (приемлемо)

Когда использовать **REST**?

- В условиях ограниченной пропускной способности соединения
- В условиях, когда необходимо кэшировать запросы
- В сервисах, предполагающих значительное масштабирование
- В сервисах, использующих AJAX

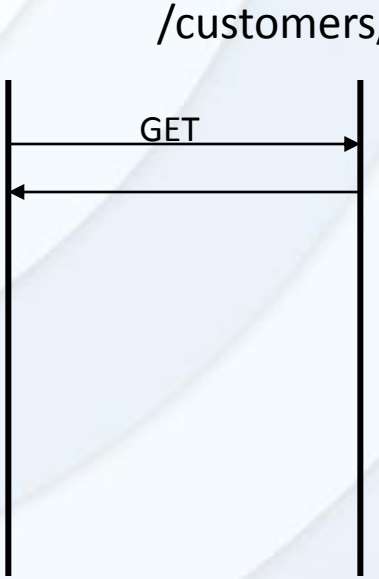
Почему не все используют **REST**?

- Нежелание нарушать W3C стандарты
- Нежелание писать user-friendly API
- Нежелание менеджеров пользоваться простым и эффективный решением
  - REST нельзя “купить”
  - REST не нужно долго учить

Как выглядит **REST**?

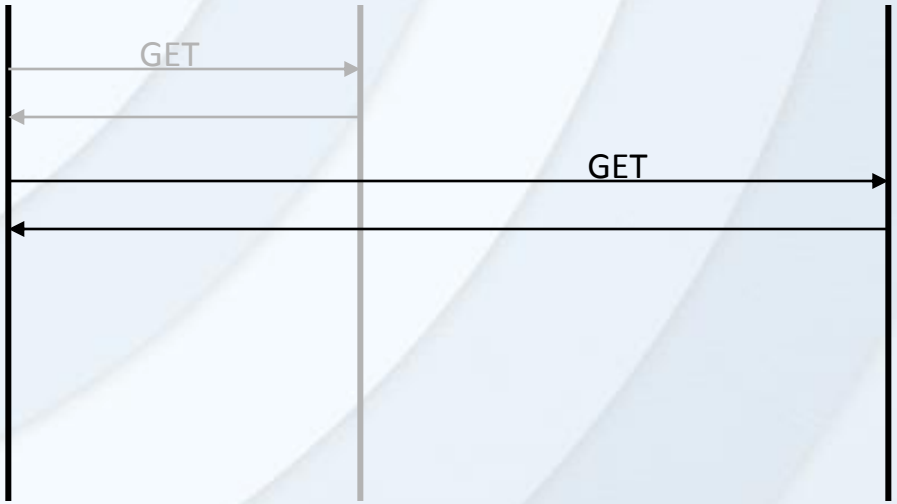
Пользователь хочет обновить свой заказ

Как выглядит **REST**?



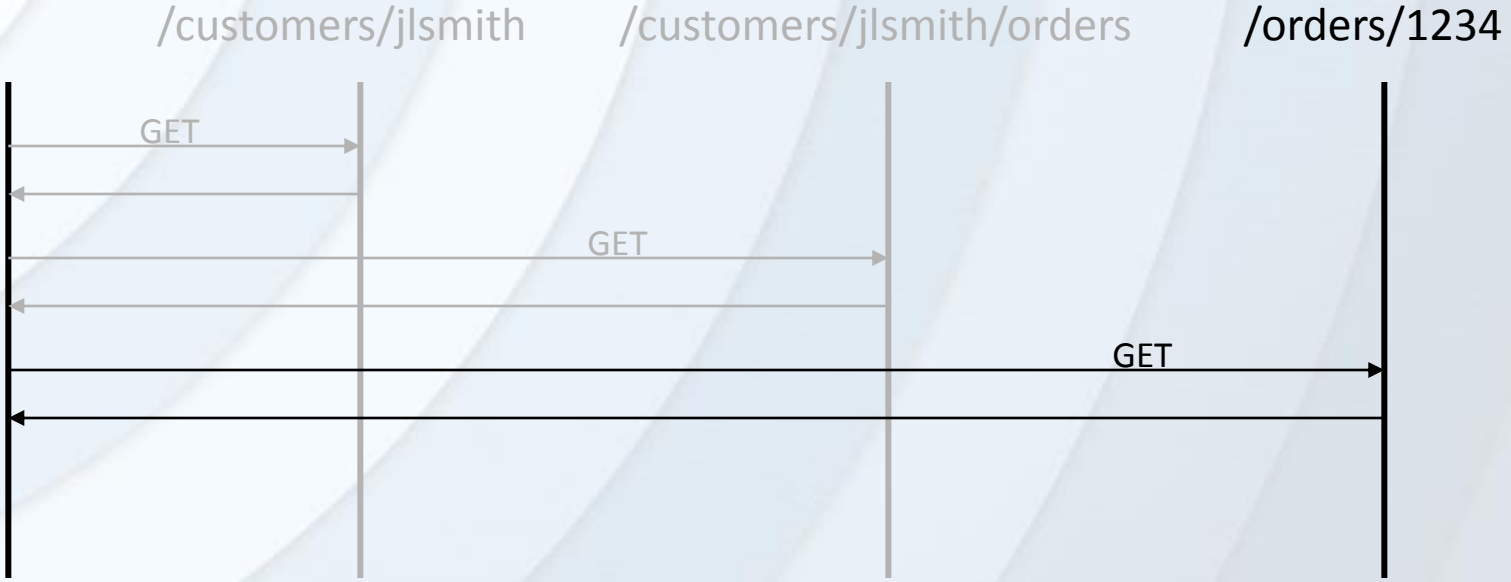

```
<customer>
        <name>James L. Smith</name>
        <portrait>http://jameslsmith.com/my_portrait.png</portrait>
        <orders>http://orderservice.com/customers/jlsmith/orders</orders>
</customer>
```

Как выглядит **REST**?




```
<orders>
        <customer>http://orderservice.com/customers/jlsmith</customer>
        <next>http://orderservice.com/customers/jlsmith/orders?before=11</next>
        <order id="20">
                <uri>http://orderservice.com/orders/1234</uri>
                <status>open</status>
        </order>
        ...
</orders>
```

Как выглядит **REST**?




```
<order>
        <customer>http://orderservice.com/customers/jlsmith</customer>
        <status>open</status>
        <item quantity="1">”Building RESTful Apps”</item>
</order>
```

Как выглядит **REST**?


```
<order>
        <customer>http://orderservice.com/customers/jlsmith</customer>
        <status>open</status>
        <item quantity="4">”Building RESTful Apps”</item>
</order>
```

Как выглядит **REST**?

GET /book/ — получить список всех книг
GET /book/3/ — получить книгу номер 3
PUT /book/ — добавить книгу (данные в теле запроса)
POST /book/3 – изменить книгу (данные в теле запроса)
DELETE /book/3 – удалить книгу

Как выглядит **REST**?

Или через POST:

POST /book/ – добавить книгу (данные в теле запроса)
POST /book/3 – изменить книгу (данные в теле запроса)
POST /book/3 – удалить книгу (тело запроса пустое)

Как строится **REST API**?

Выделяются существительные*

Существительные = ресурсы (URI)

Выделяются глаголы

Глаголы = HTTP-методы

На основе связывания HTTP-методов и URI строится API

*Если ресурс сложно описать существительным, то используется глагол

Как строится **REST API**?

Пример: Twitter REST API

`GET geo/id/:place_id`	Возвращает информацию о местоположениях
`GET geo/reverse_geocode`	По заданным координатам возвращает до 20 возможных place_id
`GET geo/search`	Возвращает местоположения, которые можно указать в statuses/update
`GET geo/similar_places`	Возвращает ближайшие местоположения, имеющие сходное название
`POST geo/place`	Создает новый объект местоположения по заданным координатам

Какие есть альтернативы для **REST**?

SOAP

- Simple Object Access Protocol
- Разработан Microsoft в 1998 г.
- Альтернатива CORBA и DCOM
- Предназначался для RPC

- Является “стандартом”

SOAP?

Пример SOAP-запроса:

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice xmlns:m="http://www.example.org/stock">
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

REST или SOAP?

	REST	SOAP
Применение	Управление ресурсами	Реализация бизнес-логики
Формат данных	XML, JSON,…	XML
Кэшируемость запросов	Да	Нет
Словарь	Основан на словаре HTTP	Требует реализации
Документация	Краткая	Объемная

POX

- Plain Old XML
- Передача XML посредством HTTP без использования SOAP-конверта, в параметрах
- Использует различные HTTP-методы для одного и того же URI
- Не использует концепцию ресурсов
- Действие, вызываемое запросом, передается в параметрах URI или теле POST

Спасибо за внимание