ПРОГРАММИРОВАНИЕ В INTERNET

REST

REST
Representational State Transfer

=

(перев. передача состояния представления) архитектурный стиль взаимодействия компонентов распределенного приложения. Ключевое понятие в REST — это ресурс. Ресурс имеет состояние, и его можно получать или изменять при помощи представлений. Приложение отвечает за некоторое множество таких ресурсов. А совокупное состояние ресурсов — это и есть состояние приложения.

RESTful сервис =

сервис, построенный с учетом REST, т.е. не нарушающий накладываемые REST ограничения

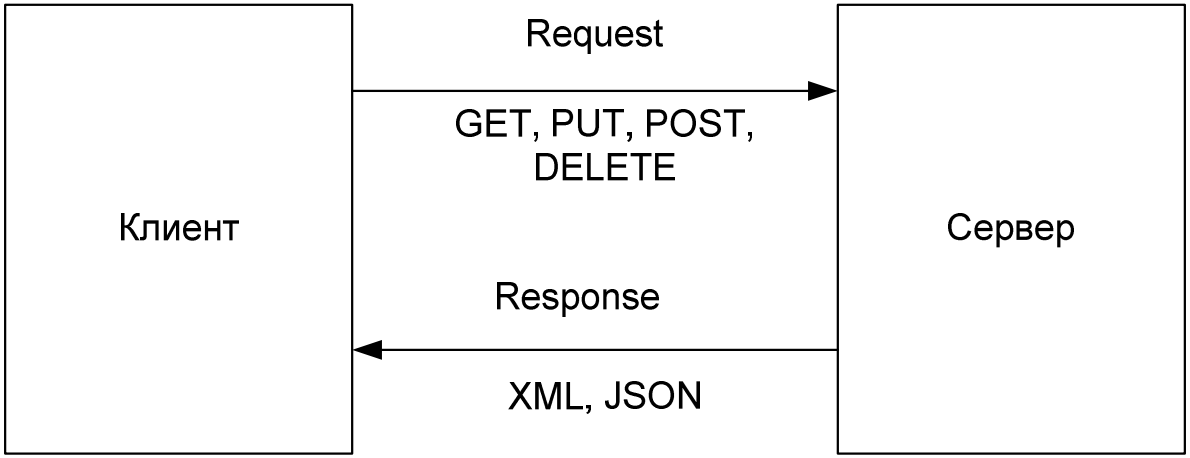
Основные характеристики

- является альтернативой RPC;
- автор Рой Филдинг, описан в диссертации «Архитектурные стили и дизайн сетевых программных архитектур», 2000 г.;
- не существует какого-то единого стандарта, который бы полностью описывал REST;
- имеет 6 ограничений;
- имеет общепринятые правила.

Ограничения REST

- модель клиент-сервер;
- отсутствие состояния на стороне сервера, сохранение состояния допускается на стороне клиента, допускается сохранение состояния в другом сервисе (например, в БД);
- кэширование на стороне клиента, сервер явно управляет кэшированием;
- единообразие интерфейсов (идентификация ресурсов, манипуляция ресурсами через представления, самодостаточные сообщения, HATEOAS);
- для клиента сервер должен представляться конечным;
- код по требованию: допускается (необязательно) выгрузка на клиент апплетов или сценариев для расширения его функциональности.

Распространенная схема RESTful сервиса



REST – это просто архитектура, он никак не привязан к каким-либо протоколам.

Наиболее распространенный протокол HTTP: GET (чтение), POST (добавление), PUT (обновление), DELETE (удаление).

Хотя REST не накладывает явных ограничений на формат, который должен быть использован для представления ресурсов, но наиболее популярными являются 2 формата: XML и JSON.

HATEOAS

Hypermedia As The Engine Of Application State

=

гипермедиа в качестве управления состоянием.

HATEOAS подразумевает, что клиенту предоставляется возможность понимать, в какое состояние он может перевести ресурс, и как он может получить эти состояния.

Гипермедиа =

технология обработки, структурирования информации и произвольного доступа к ее элементам с помощью гиперсвязей (Тед Нильсон, 1965), WWW – реализация гипермедиа.

HATEOAS (пример)

```
HTTP /1.1. 200 OK
<Студенты группа=”7-2015-2”>
<Студент  номерСтудента=1 фамилия=”Иванов”>
    <link rel=”self” href=”https://ccc/Students/7-2015-2/1”/>
    <link rel=”notes” href=”https://ccc/Students/7-2015-2/1/
notes” />
    <link rel=”rating” href=”https://ccc/Students/7-2015-2/1/
rating” />
</Студент>
<Студент  номерСтудента=2 фамилия=”Петрова”>
    <link rel=”self” href=”https://ccc/Students/7-2015-2/2”/>
    <link rel=”notes” href=”https://ccc/Students/7-2015-2/2/
notes” />
    <link rel=”rating” href=”https://ccc/Students/7-2015-2/2/
rating” />
</Студент>
</Студенты>

GET ccc/students/7-2015-2/2/rating

HTTP /1.1. 200 OK
<Рейтинг   номерСтудента = 2 значение =28 >
  <link rel=”self” href=”https://ccc/Students/7-2015-2/2/rating” />
  <link rel=”refresh” href=”https://ccc/Students/7-2015-2/2/
```

HATEOAS (пример)

```
GET /account/12345 HTTP/1.1

HTTP/1.1 200 OK
<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">100.00</balance>
    <link rel="deposit" href="/account/12345/deposit" />
    <link rel="withdraw" href="/account/12345/withdraw" />
    <link rel="transfer" href="/account/12345/transfer" />
    <link rel="close" href="/account/12345/close" />
</account>
```

```
GET /account/12345 HTTP/1.1

HTTP/1.1 200 OK
<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">-25.00</balance>
    <link rel="deposit" href="/account/12345/deposit" />
</account>
```

Общепринятые правила

- Общий префикс для всех ресурсов сервиса: http://bstu.by/api/...
- Два типа ресурсов: коллекция (api/users, api/students, ...), элемент коллекции (.../api/users/238, .../api/students/ef3d26)
- Иерархическая связь: .../api/users/238/cars/aah4899
- Использовать существительные во множественном числе. Если несколько слов, то использовать kebab-case: /api/customers/33245/delivery-addresses
- Использовать HTTP статус коды, сопроводить сообщение дополнительным кодом (например 20003, 404001,...), сделать отдельный ресурс (HATEOAS link) для пояснения ошибок: http://ccc/api/errors/20003
- Подавление статуса ответа: .../api/students/ef3d26?status_code=200
- Версионность: .../api/v7/students/ef3d26 или .../api/students/ef3d26?v=7

Общепринятые правила

- Пагинация (параметры limit, offset): .../api/students?offset=10&limit=5
- Сортировка (параметр sort): .../api/students?sort=+group,+name
- Все фильтры вынести за знак вопроса: .../api/students?minbday=19980101&maxbday=20001231&gender=m
- Пользователь получает только то, что хочет (параметр select или field): /api/students?field=bday,surname,gender
- Обозначать в запросе формат сообщений (желательна поддержка нескольких форматов, рекомендуется JSON и XML, один из форматов должен быть по умолчанию): .../api/students.json?field=bday,surname,gender
- Глобальный поиск: .../api/search?q=19600107+Иванов
- Документация

Достоинства REST

- производительность (кэширование);
- надежность (отсутствие состояния);
- простота (унифицированность интерфейса, использование повсеместных стандартов,...);
- изменяемость;
- масштабируемость.

Недостатки REST

- нет общепризнанного стандарта RESTful API;
- не все браузеры поддерживают полный словарь методов (PUT, DELETE); на практике часто используется только GET и POST (insert, delete, update);
- не однозначны коды состояний.