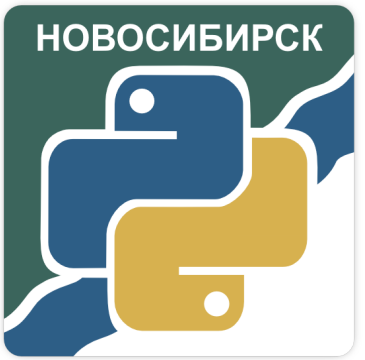
JSON-RPC

или когда REST неудобен



PyNSK #6

13.02.2016

Александр Кацко

О себе

Александр Кацко

http://alexandr.katsko.name/

О себе

Александр Кацко

http://alexandr.katsko.name/

Опыт работ с API

Опыт работ с API

Проектирование и реализация API

Использование имеющегося API

Опыт работ с API

Проектирование и реализация API

Текстовый велосипед
JSON показался излишен, команд в протоколе было минимум, потому API состоял из простых строк.

Использование имеющегося API

Опыт работ с API

Проектирование и реализация API

Текстовый велосипед
JSON показался излишен, команд в протоколе было минимум, потому API состоял из простых строк.

JSON в своём формате, отправляемый на URL. НедоREST, недоRPC.

Использование имеющегося API

Опыт работ с API

Проектирование и реализация API

Текстовый велосипед
JSON показался излишен, команд в протоколе было минимум, потому API состоял из простых строк.

JSON в своём формате, отправляемый на URL. НедоREST, недоRPC.

JSON-RPC

Использование имеющегося API

Опыт работ с API

Проектирование и реализация API

Текстовый велосипед
JSON показался излишен, команд в протоколе было минимум, потому API состоял из простых строк.

JSON в своём формате, отправляемый на URL. НедоREST, недоRPC.

JSON-RPC

REST

Использование имеющегося API

Опыт работ с API

Проектирование и реализация API

Текстовый велосипед
JSON показался излишен, команд в протоколе было минимум, потому API состоял из простых строк.

JSON в своём формате, отправляемый на URL. НедоREST, недоRPC.

JSON-RPC

REST

Использование имеющегося API

XML-RPC

Опыт работ с API

Проектирование и реализация API

Текстовый велосипед
JSON показался излишен, команд в протоколе было минимум, потому API состоял из простых строк.

JSON в своём формате, отправляемый на URL. НедоREST, недоRPC.

JSON-RPC

REST

Использование имеющегося API

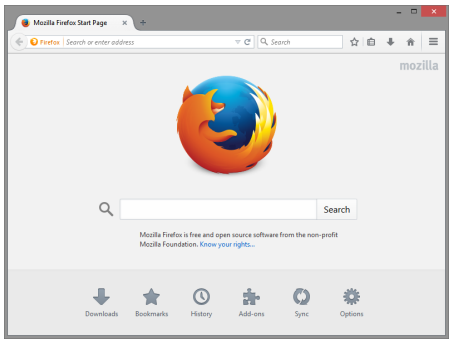
XML-RPC

Бинарные велосипеды

Опыт работ с API

Сфера применения

Браузер — сервер
HTTP, WebSockets
JSON, JSON-RPC, REST



Опыт работ с API

Сфера применения

Браузер — сервер
HTTP, WebSockets
JSON, JSON-RPC, REST

Сервер — сервер
HTTP
JSON-RPC, XML-RPC, недоREST

Опыт работ с API

Сфера применения

Браузер — сервер
HTTP, WebSockets
JSON, JSON-RPC, REST

Сервер — сервер
HTTP
JSON-RPC, XML-RPC, недоREST

Телефон — сервер
HTTP, Sockets
JSON-RPC, текстовый и бинарный велосипед

Опыт работ с API

Сфера применения

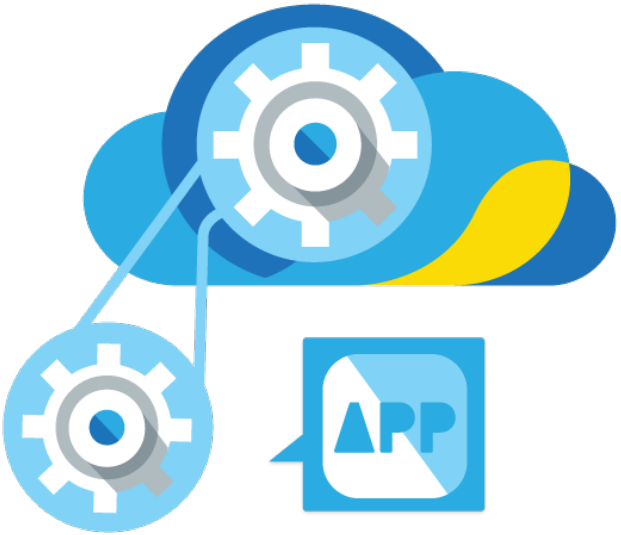
Браузер — сервер
HTTP, WebSockets
JSON, JSON-RPC, REST

Сервер — сервер
HTTP
JSON-RPC, XML-RPC, недоREST

Телефон — сервер
HTTP, Sockets
JSON-RPC, текстовый и бинарный велосипед

Десктоп — сервер — десктоп
WebSockets
JSON, JSON-RPC

Про какой API речь?

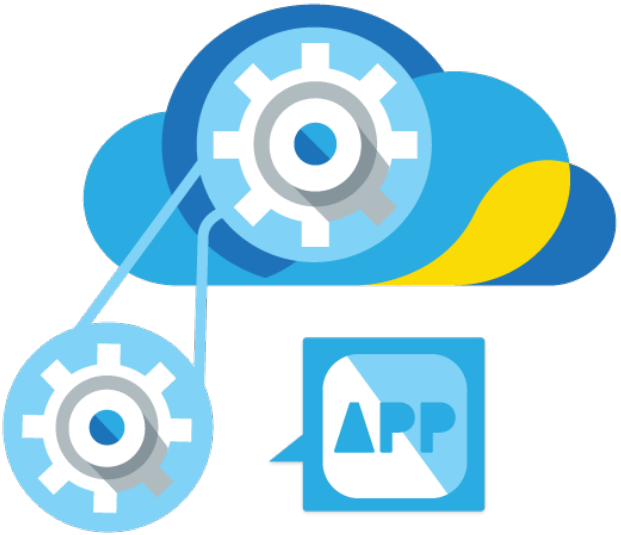


Про какой API речь?

API

Набор команд,

для использования
во внешних программах.



Два типа API

REST RPC

Два типа API

REST

RPC

Адреса (ресурсы)

Два типа API

REST

Адреса (ресурсы)

RPC

Функции

Два типа API

REST

Адреса (ресурсы)

RPC

Функции

XML-RPC

SOAP

Thrift

JSON-RPC

Два типа API

REST

Адреса (ресурсы)

RPC

Функции

XML-RPC

SOAP

Thrift

JSON-RPC

www.jsonrpc.org/specification Текущая версия **2.0 (Updated)** 04.01.2013

JSON-RPC v2.0

Формат данных

Транспорт

JSON-RPC v2.0

Формат данных — JSON

Это логотип JSON

Транспорт

JSON-RPC v2.0

Формат данных — JSON

www.json.org/json-ru.html

Это логотип JSON

Транспорт

JSON-RPC v2.0

Формат данных — JSON

www.json.org/json-ru.html

легко читается человеком, легко парсится машиной

Это логотип JSON

Транспорт

JSON-RPC v2.0

Формат данных — JSON

www.json.org/json-ru.html

легко читается человеком, легко парсится машиной

Это логотип JSON

для работы с json есть много библиотек

Транспорт

JSON-RPC v2.0

Формат данных — JSON

Это логотип JSON

www.json.org/json-ru.html

легко читается человеком, легко парсится машиной

для работы с json есть много библиотек

а если под какой-то язык её нет, то её легко сделать, т.к. json - это текст

Транспорт

JSON-RPC v2.0

Формат данных — JSON

Это логотип JSON

www.json.org/json-ru.html

легко читается человеком, легко парсится машиной

для работы с json есть много библиотек

а если под какой-то язык её нет, то её легко сделать, т.к. json - это текст

Транспорт — любой

JSON-RPC v2.0

Формат данных — JSON

Это логотип JSON

www.json.org/json-ru.html

легко читается человеком, легко парсится машиной

для работы с json есть много библиотек

а если под какой-то язык её нет, то её легко сделать, т.к. json - это текст

Транспорт — любой

HTTP, Socket, WebSocket, электронная почта,
FTP, IP посредством почтовых голубей

JSON-RPC v2.0

Доставляем JSON-RPC

Транспорт — любой

HTTP, Socket, WebSocket, электронная почта,
FTP, IP посредством почтовых голубей

JSON-RPC v2.0

Формат данных — JSON

www.json.org/json-ru.html

легко читается человеком, легко парсится машиной

Это логотип JSON

для работы с json есть много библиотек

а если под какой-то язык её нет, то её легко сделать, т.к. json - это текст

Простой формат и любой транспорт — первая киллер-фича

Транспорт — любой

HTTP, Socket, WebSocket, электронная почта,
FTP, IP посредством почтовых голубей

Структура JSON-RPC v2.0

Структура JSON-RPC v2.0

Запрос

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

Структура JSON-RPC v2.0

Запрос

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

Ответ

```
{"jsonrpc": "2.0", "result": 19, "id": 1}
```

Структура JSON-RPC v2.0

Запрос

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

Структура JSON-RPC v2.0

Запрос

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

jsonrpc - версия протокола

Структура JSON-RPC v2.0

Запрос

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

jsonrpc - версия протокола

method - имя команды

Структура JSON-RPC v2.0

Запрос

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

jsonrpc - версия протокола

method - имя команды

params - параметры для команды
(список, словарь или вообще это поле может отсутствовать)

Структура JSON-RPC v2.0

Запрос

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

jsonrpc - версия протокола

method - имя команды

params - параметры для команды
(список, словарь или вообще это поле может отсутствовать)

id - номер запроса,
чтобы знать на какой запрос пришёл ответ в случаи асинхронности

Структура JSON-RPC v2.0

Запрос

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

Ответ

```
{"jsonrpc": "2.0", "result": 19, "id": 1}
```

result - результат запрос (приметив, список, словарь)

Структура JSON-RPC v2.0

Запрос

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

Ответ

```
{"jsonrpc": "2.0", "result": 19, "id": 1}
```

result - результат запрос (приметив, список, словарь)

error - возвращаемое поле в случаи ошибки

Структура JSON-RPC v2.0

Запрос

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

Ответ

```
{"jsonrpc": "2.0", "result": 19, "id": 1}
```

Ответ с ошибкой

```
{"jsonrpc": "2.0", "error": {
     "code": 123, "message": "Invalid params",
     "data": {"text": "Params must be less then 30", "param": 42}
     },
 "id": "1"
}
```

Структура JSON-RPC v2.0

Запрос

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

Ответ

```
{"jsonrpc": "2.0", "result": 19, "id": 1}
```

Ответ с ошибкой

```
{"jsonrpc": "2.0", "error": {
     "code": 123, "message": "Invalid params",
     "data": {"text": "Params must be less then 30", "param": 42}
     },
 "id": "1"
}
```

code - номер ошибки (для удобства клиента)

Структура JSON-RPC v2.0

Запрос

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

Ответ

```
{"jsonrpc": "2.0", "result": 19, "id": 1}
```

Ответ с ошибкой

```
{"jsonrpc": "2.0", "error": {
     "code": 123, "message": "Invalid params",
     "data": {"text": "Params must be less then 30", "param": 42}
     },
 "id": "1"
}
```

code - номер ошибки (для удобства клиента)

message - краткое обозначение ошибки (строка)

Структура JSON-RPC v2.0

Запрос

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

Ответ

```
{"jsonrpc": "2.0", "result": 19, "id": 1}
```

Ответ с ошибкой

```
{"jsonrpc": "2.0", "error": {
     "code": 123, "message": "Invalid params",
     "data": {"text": "Params must be less then 30", "param": 42}
     },
 "id": "1"
}
```

code - номер ошибки (для удобства клиента)

message - краткое обозначение ошибки (строка)

data - необязательный объект с подробным описанием ошибки

Структура JSON-RPC v2.0

Запрос без id

```
{"jsonrpc": "2.0", "method": "alive"}
```

Структура JSON-RPC v2.0

Запрос без id

```
{"jsonrpc": "2.0", "method": "alive"}
```

является уведомлением и не требует ответа

Структура JSON-RPC v2.0

Пакетный запрос

```
[
 {"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1},
 {"jsonrpc": "2.0", "method": "sum", "params": [1,2,4], "id": 2}
]
```

можно отправлять в одном HTTP-запросе

Структура JSON-RPC v2.0

Пакетный запрос

```
[
 {"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1},
 {"jsonrpc": "2.0", "method": "sum", "params": [1,2,4], "id": 2}
]
```

можно отправлять в одном HTTP-запросе

Ответ

```
[
 {"jsonrpc": "2.0", "result": 19, "id": 1},
 {"jsonrpc": "2.0", "result": 7, "id": 2},
]
```

Структура JSON-RPC v2.0

Запрос

```
{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
```

Очень простая спецификация — вторая киллер-фича

Ответ

```
{"jsonrpc": "2.0", "result": 19, "id": 1}
```

Интеграция с языками программирования

Интеграция с языками программирования

Т.к. это **RPC**,
то легко переложить его запросы на язык программирования,
будь то **Python**, **JavaScript** или другой императивный язык.

Интеграция с языками программирования

Т.к. это **RPC**,
то легко переложить его запросы на язык программирования,
будь то **Python**, **JavaScript** или другой императивный язык.

```
{"jsonrpc": "2.0", "method": "approve", "params": [23, 42, 3], "id": 1}
```

Интеграция с языками программирования

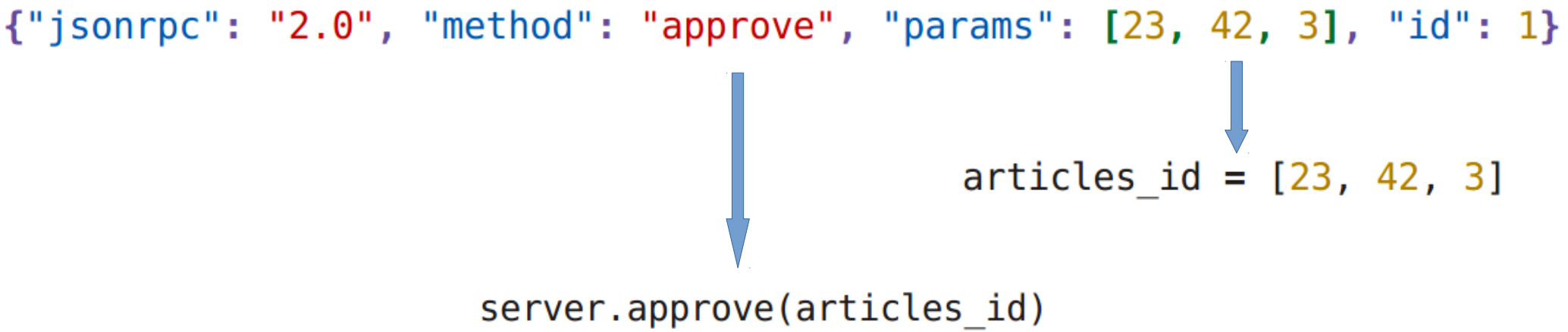
Т.к. это **RPC**,
то легко переложить его запросы на язык программирования,
будь то **Python**, **JavaScript** или другой императивный язык.

```
{"jsonrpc": "2.0", "method": "approve", "params": [23, 42, 3], "id": 1}
```

```
articles_id = [23, 42, 3]
```

Интеграция с языками программирования

Т.к. это **RPC**,
то легко переложить его запросы на язык программирования,
будь то **Python**, **JavaScript** или другой императивный язык.



Выглядит почти одинаково как для Python, так и для JavaScript :)

Интеграция с языками программирования

Пример запроса с использованием AngularJS
и планига к нему github.com/ajsd/**angular-jsonrpc**

Интеграция с языками программирования

Пример запроса с использованием AngularJS
и планига к нему github.com/ajsd/**angular-jsonrpc**

```
// Настройка: указание адреса сервера и объявление метода

var app = angular.module("app", ["jsonrpc"])
    .config(function(jsonrpcProvider){
        jsonrpcProvider.setBasePath("/api/");
    })
    .service('server', function(jsonrpc) {
        var service = jsonrpc.newService('article');
        this.approve = service.createMethod('approve');
    });

// Контроллер

app.controller("articlesApprove", function($scope, server) {
    var articlesId = [23, 42, 3];
    server.approve(articlesId)
        .then(function(data) {
            $scope.data = data;
        })
        .catch(function(data) {
            console.log(data);
        });
});
```

Интеграция с языками программирования

Пример запроса с использованием AngularJS
и планига к нему github.com/ajsd/**angular-jsonrpc**

```
15      var articlesId = [23, 42, 3];
16      server.approve(articlesId)
17          .then(function(data) {
18              $scope.data = data;
19          })
20          .catch(function(data) {
21              console.log(data);
22          });
```

Интеграция с языками программирования

Обработка запроса на **Django**

Интеграция с языками программирования

Обработка запроса на **Django**

(используется библиотека github.com/samuraisam/**django-json-rpc**)

```
 1 # urls.py
 2 from jsonrpc import jsonrpc_site
 3
 4 urlpatterns = [
 5     url(r'^api/', jsonrpc_site.dispatch, name="api"),
 6 ]
 7
 8 # views.py
 9 from jsonrpc import jsonrpc_method
10 from article.models import Article
11
12 @jsonrpc_method('article.approve')
13 def approve(request, ids):
14     result = []
15     articles = Article.objects.filter(pk__in=ids)
16     for article in articles:
17         article.approve()
18         result.append(article.id)
19     return result  # {"jsonrpc": "2.0", "result": [42, 3], "id": 1}
```

Интеграция с языками программирования

Пример отправки запроса из **Django**

(используется та же библиотека)

```
1 from jsonrpc.proxy import ServiceProxy
2 s = ServiceProxy('http://localhost:8000/api/')
3 result = s.article.approve([23, 42, 3])
```

Интеграция с языками программирования

Пример отправки запроса из **Django**

(используется та же библиотека)

```
1 from jsonrpc.proxy import ServiceProxy
2 s = ServiceProxy('http://localhost:8000/api/')
3 result = s.article.approve([23, 42, 3])
```

Лёгкое нативное использование из кода программы — третья киллер-фича

Отладка JSON-RPC

Если требуется выполнить запрос не из своего кода...

Отладка JSON-RPC

Если требуется выполнить запрос не из своего кода...

CURL

Отладка JSON-RPC

Если требуется выполнить запрос не из своего кода...

CURL

Графические HTTP-клиенты

Отладка JSON-RPC

Если требуется выполнить запрос не из своего кода...

CURL

Графические HTTP-клиенты

Дополнения к браузерам

Отладка JSON-RPC

Если требуется выполнить запрос не из своего кода...

CURL

Графические HTTP-клиенты

Дополнения к браузерам

Например, **Postman**
дополнение для Chrome с подсветкой JSON и историей запросов
https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop

Где стоит использовать JSON-RPC

Где стоит использовать JSON-RPC

Почти везде :)

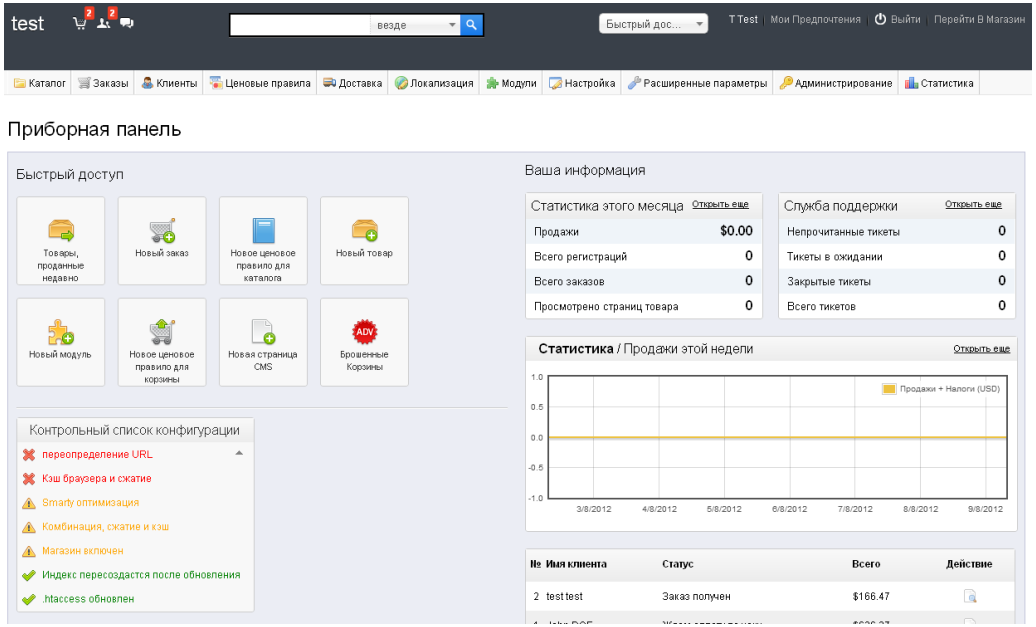
Где стоит использовать JSON-RPC

Почти везде :)

где не требуется очень быстрая обработка,
т.к. для этого **текстового JSON** уже мало

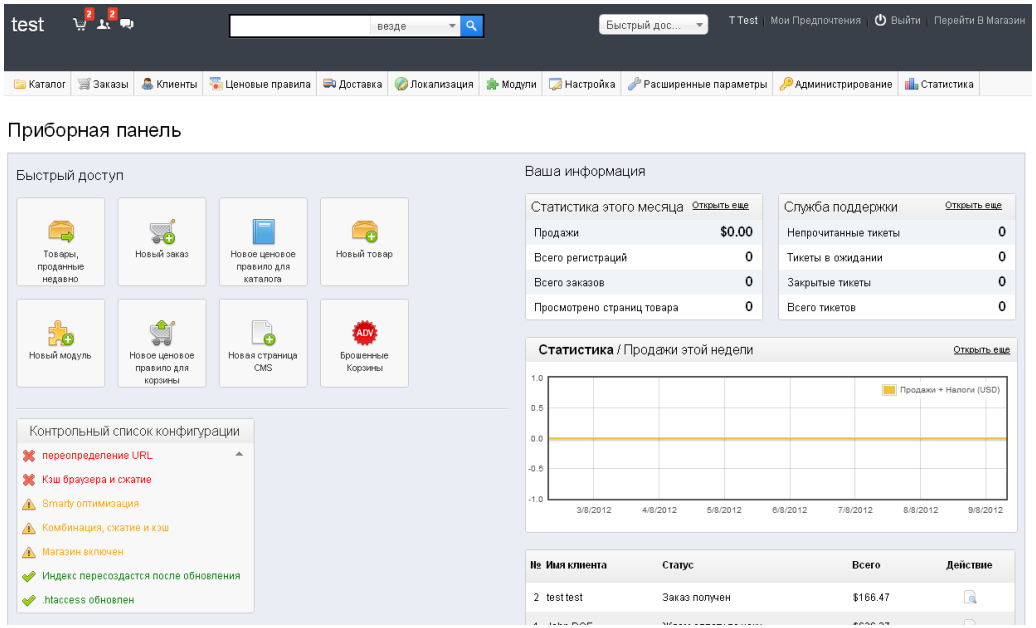
Где стоит использовать JSON-RPC

Очень удобно использовать в админках



Где стоит использовать JSON-RPC

Очень удобно использовать в админках



Не надо придумывать к кому бы адресу обратится и каким HTTP-методом

Просто пишем функции как в привычном языке программирования

Где стоит использовать JSON-RPC

Удобно использовать в мобильных клиентах

Где стоит использовать JSON-RPC

Удобно использовать в мобильных клиентах

И не упираться в ограничение на использование только HTTP

Использовать WebSockets и Sockets

А где же применять REST

А где же применять REST

RESTful API
GET PUT POST DELETE

Там, где вся логика работы укладывается в CRUD и HTTP

А где же применять REST

RESTful API
GET PUT POST DELETE

Там, где вся логика работы укладывается в CRUD и HTTP

И есть уверенность, что с развитием проекта так оно и останется

А где же применять REST

RESTful API
GET PUT POST DELETE

Там, где вся логика работы укладывается в CRUD и HTTP

И есть уверенность, что с развитием проекта так оно и останется

Чтобы не ограничиваться CRUD и HTTP стоит посмотреть в сторону JSON-RPC

Он такой же простой в написании и чтении, как REST, но не имеет его ограничений.

Если нужны гигантские скорости обработки

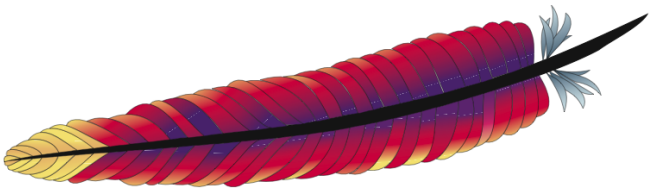
Если нужны гигантские скорости обработки

То нужно смотреть в сторону бинарных протоколов и форматов

Если нужны гигантские скорости обработки

То нужно смотреть в сторону бинарных протоколов и форматов

Пример протокола — **Thrift**
thrift.apache.org



Apache Thrift ™

Если нужны гигантские скорости обработки

То нужно смотреть в сторону бинарных протоколов и форматов

Пример протокола — **Thrift**
thrift.apache.org

Пример формата — **Protocol Buffers**
developers.google.com/protocol-buffers/

Если нужны гигантские скорости обработки

То нужно смотреть в сторону бинарных протоколов и форматов

Пример протокола — **Thrift**
thrift.apache.org

Пример формата — **Protocol Buffers**
developers.google.com/protocol-buffers/

Их плюс в скорости работы.

Минус — **они сложнее** в использовании и отладке, чем JSON-RPC.