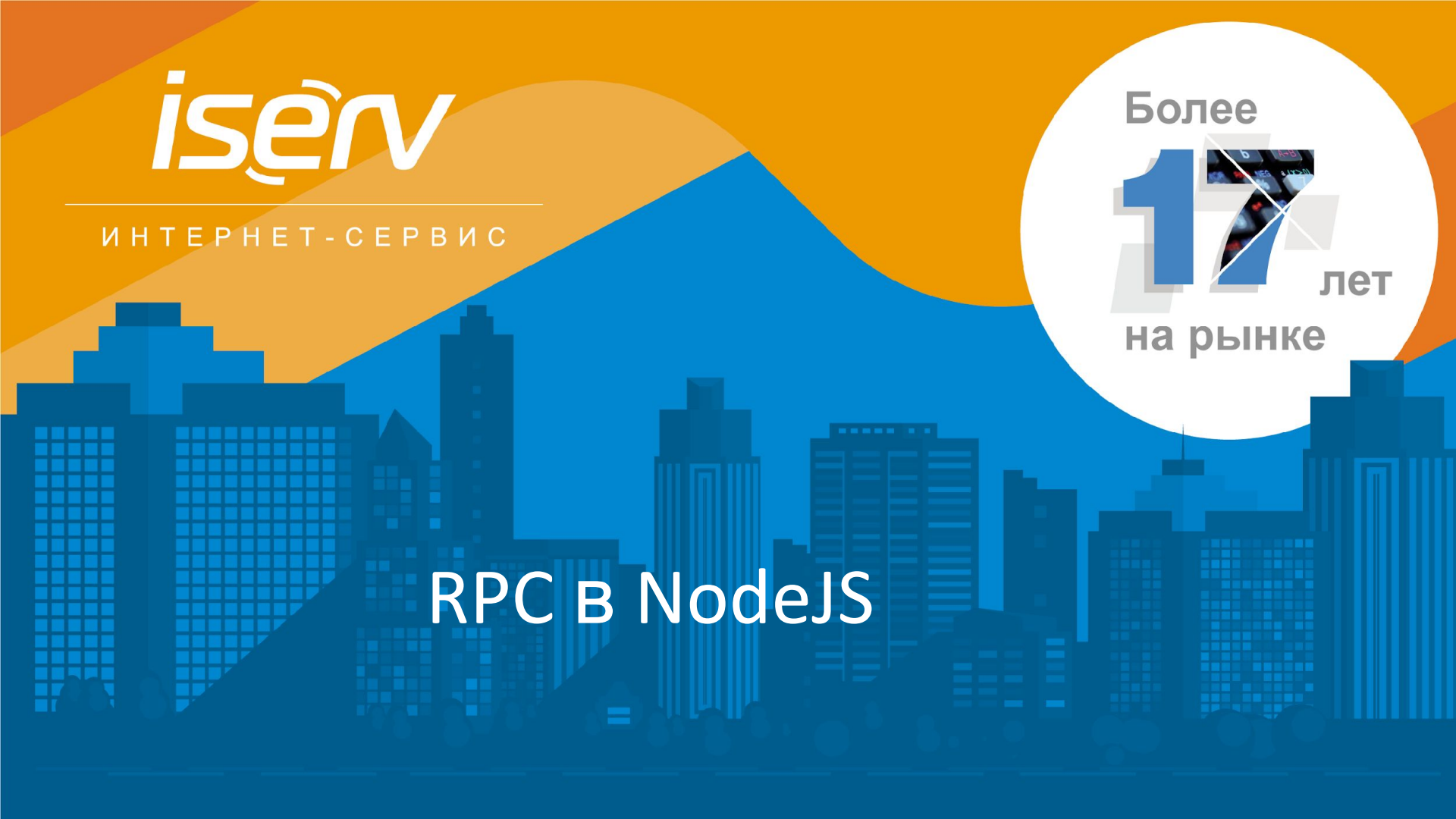


RPC в NodeJS

RPC - *Remote Procedure Call*

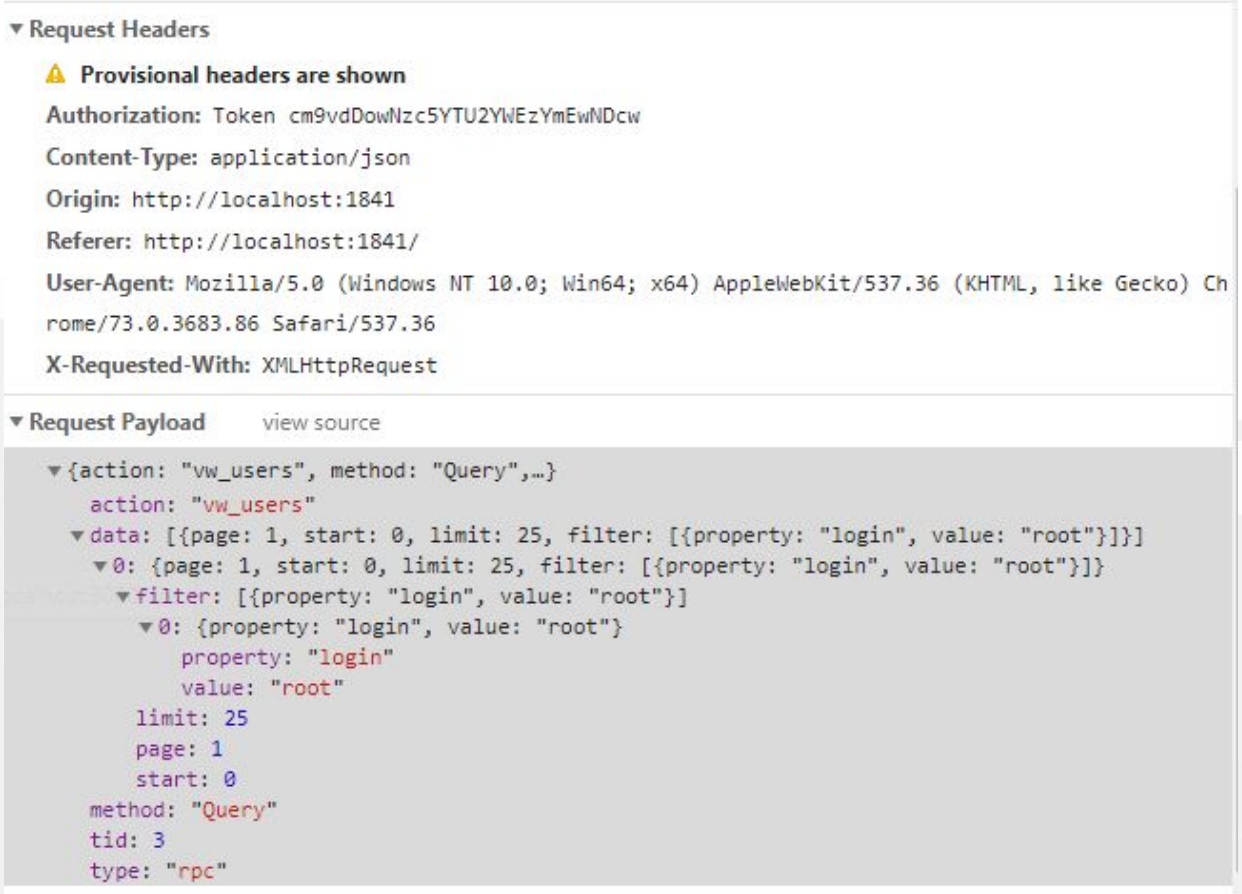
- action: string – имя сущности
- method: string – вызываемый метод сущности
- data: any[] - данные
- type: string – тип операции. По умолчанию “rpc”
- tid: number – идентификатор запроса

Пример RPC запроса

```
[
    {
        “action”: “users”,
        “method”: “Query”,
        “data”: [{
            “filter”: [{
                “property”: “login”,
                “value”: “user”
            }]
        }],
        “type”: “rpc”,
        “tid”: 1
    }
]
```

Документация:

RPC в браузере



Результат RPC

- meta
  - success: boolean – результат выполнения запроса
  - msg: string – текст ошибки, если success = false
- code:number – HTTP статус код
- result
  - records: any[]|any – массив данных или объект
  - total: number – число результатов
- time: number – время выполнения sql запроса
- sql
  - query: string – запрос к БД
- tid: number – идентификатор запроса
- type: string – тип запроса
- method: string – метод сущности
- action: string – имя сущности
- rpcTime: number – время выполнения RPC функции

Результат RPC


```
[
  {
    "meta": {
      "success": true,
      "msg": "ok"
    },
    "code": 200,
    "result": {
      "records": [
        {
          "id": 1,
          "name": "Главный",
          "login": "root",
          "claims": ".master.admin.filer.support.",
          "email": null,
          "tel": null,
          "last_sign_in": null,
          "icon_fileimage": null,
          "description": null,
          "address": null
        }
      ],
      "total": 1
    },
    "time": 31,
    "sql": {
      "query": "SELECT * FROM public.vw_users as v WHERE v.login = 'root' LIMIT 25;",
      "params": []
    },
    "tid": 3,
    "type": "rpc",
    "method": "Query",
    "action": "vw_users",
    "rpcTime": 33
  }
]
```

RPC «сущность» в NodeJS

```
exports.[имя сущности] = function(session) {
        return {
                isLocal: true, /// !!!

                [имя сущности]: function(data, callback) {
                        callback([результат]);
                }
        };
}
```

Пример RPC «сущности»

```
8   /**
9    * объект для формирования ответа
10   */
11  var result_layout = require('postgresql-rpc-dbcontext/modules/result-layout');
12  var authorizeDb = require('../modules/authorize/authorizeDb');
13  var saltHash = require('../modules/authorize/saltHash');
14
15  /**
16   * Объект с набором RPC функций
17   */
18  exports.shell = function (session) {
19
20      return {
21          isLocal: true, // нужно указывать, иначе безопасность при создании meta не пропустит
22
23          /**
24           * Получение серверного времени
25           * @param {*} data
26           * @param {*} callback
27           * @example
28           * // никаких параметров не нужно передавать
29           * PN.shell.getServerTime({}, function(){})
30           */
31          getServerTime: function (data, callback) {
32              callback(result_layout.ok([new Date()]));
33          },
```

Регистрация созданной сущности в RPC

```
var shell = require('postgresql-rpc-shell');

var shellObject = {
        db: db,
        namespace: ns,
        contexts: [
                require('./modules/webapi'),
                require('./modules/adminapi')
        ],
        …
};

app.use(shell(shellObject));
```

postgresql-rpc-shell

Основная библиотека для работы с RPC в NodeJS

- Авторизация
- Обработка запросов
- Выполнение операции
- Фильтрация данных по безопасности

Документирование:
https://docs.appcode.pw/projects?project=postgresql-rpc-shell

postgresql-rpc-shell

Основная библиотека для работы с RPC в NodeJS

- Авторизация
- Обработка запросов
- Выполнение операции
- Фильтрация данных по безопасности

Документирование:
https://docs.appcode.pw/projects?project=postgresql-rpc-shell

Взаимодействие с БД


```
…
var render = require('dynamic-schema-reader'); // модуль генерации шаблонов

module.exports = function (callback) {
        reader({
                …
                        connectionString: connectionString
                        autoRemove: true,
                        schemaList: ["'core'", "'public'", ...]
        }, function (schemas) {
                var content = ejs.compile(…);
                fs.writeFileSync(path, content);
                …
        });
}
```

Пример dbcontext.js

```
exports.chat = function (session) {
    return {
        Query: function (query_param, callback) {
            provider.select('public', 'chat', query_param, filter.security(session), callback);
        },
        Add: function (data, callback) {
            provider.insert('public', 'chat', data, callback);
        },
        Update: function (data, callback) {
            provider.update('public', 'chat', 'id', data, callback);
        },
        Delete: function (data, callback) {
            provider.delete('public', 'chat', 'id', data, callback);
        },
        Count: function (query_param, callback) {
            provider.count('public', 'chat', query_param, callback);
        }
    }
}
```

postgresql-rpc-dbcontext

Основная библиотека для работы БД Postgresql

- Выполнение CRUD операций с таблицами
- Вызов представлений и функций

Документирование:
https://docs.appcode.pw/projects?project=postgresql-rpc-dbcontext

Примеры

- Создание собственной RPC функции и ее подключение
  - Способы возвращения результатов
- Добавление новой таблицы и обработка ее через RPC
- Просмотр данных в интерфейсе ExtJS
  - Интерфейс тестирования запросов
- Добавление безопасности, фильтрация данных
- Работа с websocket (https://www.appcode.pw/?p=1256)

iserv

ИНТЕРНЕТ-СЕРВИС

Спасибо за внимание!