lab-04 Создание простых http запросов (продолжение lab-03)

В современных условиях часто приходится работать с разными http сервисами. Т.е. наше приложение выступает агрегатором, которое использует внешние сервисы. Для более глубокого понимания как работаю callback вызова, сделаем в даной лабораторной работе примеры использования внешних API.

Для создания внешних вызово используем библтотеку: SuperAgent

Линки на страницу API NBU по получению курсов НБУ

Страница всех API НБУ находится по линку: страница API НБУ. Также, более детальная документация может быть получена на линку: скачается pdf

Согласно данных документов:

- курс НБУ на заданную дату по всем валютам может быть получен http-get запросом: https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?date=20201101

Курс на дату (задається у форматі YYYYMMDD, де YYYY - рік, MM - місяць, DD - день):

Курс НБУ на заданную дату по всем валютам может быть получен http-get запросом: https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?valcode=EUR&date=20201101

Пробуем написать сервис для подучения курсов за период

На вход сервис принимает такие реквизиты:

- дата начала периода
- дата окончания периода

Обработка запроса: Сервис вызывает API НБУ по получению курсов и каждую дату сохраняет в свой файл

Основной обработчик сервиса находится в модуле nbu-exch-srvc-dsdf.js

Для быстрого получения курса за дату написан клон функции getExchRateByDate: функция **getExchRateByDateP** в виде Promise. А в **./test/test-nbu-exch-srvc-dsdf.js** - написан тестовый кейс для этой фукнции. Ниже показан пример тестового кейса с вызовом функции, которая возвращает Promise **getExchRateByDateP**. Сравните с вызовом в lab-03 тестового кейса для функции **getExchRateByDate**.

```
    it('function getExchRateByDateP - ожидаем курсы валют за дату', function(done){
          // заглушка - пропуск тестового кейса
          this.skip();
          // url=https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?
date=20201101&json
          var l_param = { exchdate: '2020-10-01' } ;
          srvc.getExchRateByDateP( l_param )
          .then (result => {
```

```
                console.log( JSON.stringify(  result  )  );

                // Пример ответа: {"ok":true,"filename":"exch20201001.json"}
                // проверяю наличие полей в ответе
                res.body.should.have.property('ok');
                res.body.should.have.property('filename');

                // проверяю значение поля "ok"
                assert.typeOf( res.body.ok, 'boolean');
                expect(res.body.ok).to.equal( true );
                // проверяю значение поля "filename"
                assert.typeOf( res.body.filename, 'string');
                done();

            })
            .catch(err => {
              console.log(err.message);
              done(err);
            })
        }); //it
```

Нужно отменить, что конструкций **.then(result => {.....})** может быть целая цепочка.

Теперь, так как нам нужно получить курсы за период воспльзуется шаблоном Promise.all, который в параллель вызовет сразу несколько сервисов и вернет массив результатов. Если хоть один сервис вернет ошибку - то и результат будет ошибочным. Для этого написна функция:

```
/**
 * Получение курсов за период
 * @param {object} a_param
 *    @param {string} exchds дата начала периода в формате YYYY-MM-DD
 *    @param {string} exchdf дата окончания периода в формате YYYY-MM-DD
 *  Пример:
 *  a_param = {exchds: '2020-11-01', exchdf: '2020-11-01'}
 */
function getExchRateByRngP ( a_param )
```

Ключевимы элементами этого шаблона есть: -- массив вызовов:

```
  // массив промисов на выполннение
            var EachPromise = [];
```

- inline функция

```
    var l_proc = function( p_param, p_that ){
        return getExchRateByDateP( p_param, p_that);
```

```
    };
```

- цикл с накачкой массива вызовами фунции:

```
            // итерирую по массиву дат от первой даты до последней
            // и накачиваю массив EachPromise вызовами inline-функции
            for (var d = l_dts; d <= l_dtf; d.setDate(d.getDate() + 1)) {
                var l_cdt = new Date(d);
                console.log(   l_cdt.toISOString().slice(0, 10) );
                var l_param = { exchdate: l_cdt.toISOString().slice(0, 10) };
                EachPromise.push(  l_proc( l_param, that ) );
            }
```

-- Непосредственно вызов promise.all и возврат результата или ошибки

```
    // Вызываю  Promise.all
            Promise.all(EachPromise)
            .then( results => {
                // При успешной обработке  получим массив результатов
                // его и вернем в параметре "list:"
                resolve( {  ok: true, list: results} );
            })
            .catch( err => {
                // тут возврат  ошибки
                reject(err);
            })
```

- пример вызова в тестовм кейсе

```
 it('function getExchRateByRngP - ожидаем курсы валют за период', function(done){
        // заглушка - пропуск тестового кейса
        //this.skip();
        // url=https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?
date=20201101&json
        var l_param = { exchds: '2020-10-01', exchdf: '2020-10-11' } ;
        srvc.getExchRateByRngP( l_param )
        .then (result => {
              console.log( JSON.stringify(  result  )  );

              // Пример ответа: {"ok":true,"filename":"exch20201001.json"}
              // проверяю наличие полей в ответе
              //res.body.should.have.property('ok');
              //res.body.should.have.property('filename');

              // проверяю значение поля "ok"
              //assert.typeOf( res.body.ok, 'boolean');
```

```
                //expect(res.body.ok).to.equal( true );
                // проверяю значение поля "filename"
                //assert.typeOf( res.body.filename, 'string');
                done();

        })
        .catch(err => {
          console.log(err.message);
          done(err);
        })
    }); //it
```

Ну и смотрим сам результат:

```
{
    "ok": true,
    "list": [
        {
            "ok": true,
            "filename": "exch20201001.json"
        },
        {
            "ok": true,
            "filename": "exch20201002.json"
        },
        {
            "ok": true,
            "filename": "exch20201003.json"
        },
        {
            "ok": true,
            "filename": "exch20201004.json"
        },
        {
            "ok": true,
            "filename": "exch20201005.json"
        },
        {
            "ok": true,
            "filename": "exch20201006.json"
        },
        {
            "ok": true,
            "filename": "exch20201007.json"
        },
        {
            "ok": true,
            "filename": "exch20201008.json"
        },
        {
            "ok": true,
```

```
            "filename": "exch20201009.json"
        },
        {
            "ok": true,
            "filename": "exch20201010.json"
        },
        {
            "ok": true,
            "filename": "exch20201011.json"
        }
    ]
}
```

Http роутер не написан. Можете написать сами. Ну и ответ как-то тнасформироать нужно. слишком много **ok:true**. И, посомтрите сколько файлов записалось в каталог exchrates