

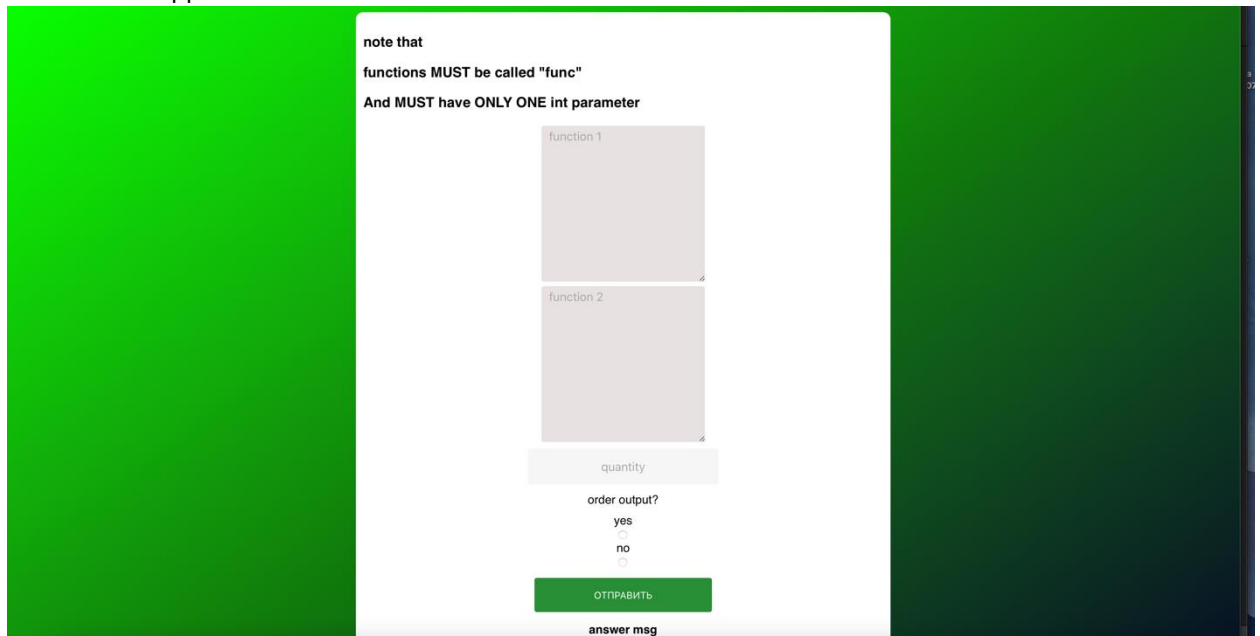
## Отчет о проделанной работе:

Целевое приложение server находится в папке backendTitan, небольшая репрезентация – клиент fronttitan – react приложение

Реализовал Spring Boot + WebFlux приложение (калькулятор), которое асинхронно обрабатывает запросы с клиента, неблокирующееся, масштабируемое, event-driven, предусмотрена **серверсайдная валидация, обработка исключений**, 2 режима mode=true (ordered – упорядоченная выдача результатов) и mode=false (unordered – неупорядоченная), предусмотрено сообщение клиенту о том, что данные введены некорректно, на стороне сервера **нет буферизации**, отказоустойчивость тоже есть, предусмотрен случай с  $\geq 1000$  значений (миллион и тд.), для наглядности ниже пример работы – postman и fronttitan

## Пример работы

Внешний вид:



The screenshot shows a web application interface with a white central panel on a dark green background. At the top, it says "note that functions MUST be called 'func'" and "And MUST have ONLY ONE int parameter". Below this are two text input fields labeled "function 1" and "function 2". Underneath these is a "quantity" input field. Then there is a question "order output?" with two radio button options: "yes" and "no". At the bottom of the form is a green button labeled "ОТПРАВИТЬ". Below the button, the text "answer msg" is visible.

note that

functions **MUST** be called "func"

And **MUST** have **ONLY ONE** int parameter

```
function func(j){  
  return j*998;  
}
```

```
function func(j){  
  var r = -65000;  
  var t = 0;  
  while(r<65000){  
    r++;  
    t=0;  
    while(t<100){  
      t++;  
    }  
  }  
  return j;  
}
```

1000

order output?

yes

☐

no

☒

ОТПРАВИТЬ

answer

msg

Начну с описания input:

Func1 – просто возвращает значение умноженное на константу

Func2 – возвращает значение после прохождения 2 вложенных циклов, чтобы значительно замедлить выполнение

Количество итераций 1000, заодно проверим падает или нет при большом количестве, для этих примеров использовался интервал времени в 1 нс, чтобы мне не сидеть и не ждать около часа

ОТПРАВИТЬ

| answer                      | msg     |
|-----------------------------|---------|
| <0>,<1>,<0.0>,<4431337>     | Success |
| <1>,<1>,<998.0>,<2494125>   | Success |
| <2>,<1>,<1996.0>,<2532788>  | Success |
| <3>,<1>,<2994.0>,<2685324>  | Success |
| <0>,<2>,<0>,<23504991>      | Success |
| <4>,<1>,<3992.0>,<2419336>  | Success |
| <5>,<1>,<4990.0>,<2689399>  | Success |
| <6>,<1>,<5988.0>,<2061540>  | Success |
| <7>,<1>,<6986.0>,<1873866>  | Success |
| <8>,<1>,<7984.0>,<1731853>  | Success |
| <9>,<1>,<8982.0>,<1725269>  | Success |
| <10>,<1>,<9980.0>,<2448883> | Success |

Результат работы при режиме обычной неотсортированной выдаче, видим, что в поле msg успешное прохождение валидаторов ( функции func1 и func2 – валидны), видим, что пятое значение – это результат выполнения второй функции, то есть в среднем 4-5 значений первой функции приходят пока выполняется вторая

order output?

yes

no

ОТПРАВИТЬ

| answer  | msg             |
|---|-----------------|
| <0>,<0>,<3629615>,<4>,<0>,<0>,<23618915>,<0>      | Success Success |
| <1>,<1>,<2887125>,<9>,<1>,<1>,<21252173>,<0>      | Success Success |
| <2>,<2>,<2913756>,<15>,<2>,<2>,<21110095>,<0>     | Success Success |
| <3>,<3>,<2914056>,<21>,<3>,<3>,<20554681>,<0>     | Success Success |
| <4>,<4>,<2816824>,<27>,<4>,<4>,<21897473>,<0>     | Success Success |
| <5>,<5>,<2316448>,<26>,<5>,<5>,<18854340>,<0>     | Success Success |
| <6>,<6>,<2174274>,<25>,<6>,<6>,<18691597>,<0>     | Success Success |
| <7>,<7>,<2742977>,<24>,<7>,<7>,<18197937>,<0>     | Success Success |
| <8>,<8>,<2353549>,<23>,<8>,<8>,<19558872>,<0>     | Success Success |
| <9>,<9>,<2250808>,<22>,<9>,<9>,<17874605>,<0>     | Success Success |
| <10>,<10>,<2458098>,<21>,<10>,<10>,<17362578>,<0> | Success Success |
| <11>,<11>,<2677817>,<20>,<11>,<11>,<17480322>,<0> | Success Success |
| <12>,<12>,<2030846>,<19>,<12>,<12>,<17332528>,<0> | Success Success |
| <13>,<13>,<1525862>,<18>,<13>,<13>,<16965176>,<0> | Success Success |
| <14>,<14>,<2033370>,<17>,<14>,<14>,<17059477>,<0> | Success Success |
| <15>,<15>,<1583886>,<16>,<15>,<15>,<15799520>,<0> | Success Success |

Результат работы при режиме отсортированной выдаче, возьмем первое, мы видим, как и в прошлый раз что успело выполниться сразу 4 функции 1 пока выполнялась функция 2, далее идет прирост в количестве выполняемых функций 1

POST http://localhost:8080/calculator/calculate Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   ... "func1": "function func(inp) {return inp*88888}",
3   ... "func2": "function func(j) {var r=-65000; var t=0; while(r<65000){r++; t=0; while(t<100){t++;
4     ... "quantity": 1000,
5     ... "mode": true
6   }
```

Body Cookies Headers (5) Test Results 200 OK 31.40 s 105.11 KB Save Response

Pretty Raw Preview Visualize JSON

```
984   "msg": "Success      Success"
985   },
986   {
987     "answer": "<246>,<2.1866448E7>,<1255710>,<25>,<246>,<246>,<17516086>,<0>",
988     "msg": "Success      Success"
989   },
990   {
991     "answer": "<247>,<2.1955336E7>,<1451286>,<24>,<247>,<247>,<16956895>,<0>",
992     "msg": "Success      Success"
993   },
994   {
995     "answer": "<248>,<2.2044224E7>,<954469>,<23>,<248>,<248>,<17762081>,<0>",
996     "msg": "Success      Success"
997   },
998   {
999     "answer": "<249>,<2.2133112E7>,<699748>,<22>,<249>,<249>,<16955750>,<0>",
1000    "msg": "Success      Success"
```

Здесь представлен пример опять же отсортированного вывода

POST

http://localhost:8080/calculator/calculate

Send

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

2

...

"func1": "function func(inp) {return inp\*88888};",

3

...

"func2": "function func(j){var r=-65000; var t=0; while(r<65000){ r++; t=0; while(t<100){ t++;

4

...

quantity": 1000,

5

...

mode": false

BodyCookiesHeaders (5)Test Results200 OK17.27 s114.8 KBSave Response

Pretty

Raw

Preview

Visualize

JSON

593

}

594

{

595

"answer": "<132>,<1>,<1.1733216E7>,<886492>",

596

"msg": "Success"

597

},

598

{

599

"answer": "<133>,<1>,<1.1822104E7>,<909175>",

600

"msg": "Success"

601

},

602

{

603

"answer": "<16>,<2>,<16>,<16942481>",

604

"msg": "Success"

605

},

606

{

607

"answer": "<134>,<1>,<1.1910992E7>,<987228>",

608

"msg": "Success"

Здесь представлен пример неотсортированного вывода