Performance Analysis when adding a product to the receipt

We used Profiler and Timeline in Chrome Developer Tools to detect where the application is spending more time when adding a product to the receipt.

As a general rule, all the SQL Transactions are the most expensive transactions of the application, but we detected two main points to analyze:

- Calculate Gross with the discounts.
- Calculate Taxes for each line.

In the following Slides you can see the calculation of taxes in the timeline:

Developer Tools - http://localhost:8080/openbravo/web/org.openbravo.retail.posterminal/?terminal=VBS-1

Elements Network Sources Timeline Profiles Resources Audits Console

3 695

Stacks Memory

Filter All Loading Scripting Rendering Painting

30 fps
60 fps

2000 ms 2500 ms 3000 ms 3500 ms 4000 ms 4500 ms 5000 ms 5500 ms

- Function Call (b1f838428...
- Function Call (b1f838428...
- Function Call (b1f838428...
- Function Call (b1f838428...
- Event (mouseup)
- Event (click)
- Timer Fired (938)
- Function Call (b1f838428...
- Function Call (b1f83842...
  - Function Call (InjectedSc...
  - Function Call (InjectedSc...
  - Function Call (InjectedSc...
  - GC Event (13.8 MB collec...
- Event (mousemove)
- Event (mousemove)
- Function Call (b1f838428...
- Function Call (b1f838428...
- Function Call (b1f838428...
- Function Call (b1f838428...
- Function Call (b1f838428...

MEMORY

Used JS Heap [64555184:9168

Documents [1:1]

Function Call Details

0.088 ms

0.088 ms Scripting

Self Time: 0.088 ms

Start Time: 3.64 s

Location: InjectedScript:1

Call Stack:

processConsoleLevel @ b1f83842892ba77a76c66d366f09d937_WebPOS.js:7105
OB.warn @ b1f83842892ba77a76c66d366f09d937_WebPOS.js:7133
hookManager.executeHooks @ b1f83842892ba77a76c66d366f09d937_WebPOS.js:3154
enyo.kind.renderTotal @ b1f83842892ba77a76c66d366f09d937_WebPOS.js:24931
(anonymous function) @ b1f83842892ba77a76c66d366f09d937_WebPOS.js:25188
Backbone.Events.trigger @ backbone-0.9.2.js:163
_.extend.change @ backbone-0.9.2.js:473
_.extend.set @ backbone-0.9.2.js:314
(anonymous function) @ b1f83842892ba77a76c66d366f09d937_WebPOS.js:16228
(anonymous function) @ b1f83842892ba77a76c66d366f09d937_WebPOS.js:23755

Performance Analysis when adding a product to the receipt

As you can see there is an operation that is costing 117.265 ms:
Location: b1f83842892ba77a76c66d366f09d937_WebPOS.js:1702

After adding a line to the receipt and before rendering the total in the green payment button, the system recalculates taxes
... this.calculateTaxes
...and after calculating taxes, the hook OB.UTIL.HookManager.executeHooks('OBPOS_FindTaxRate') is executed...
...and here is where it spends a lot of time, in OBPOS_FindTaxRate hook's callback: executing this query:

OB.Dal.query(OB.Model.TaxRate, args.sql, [], function (coll, args)

See next slide to get a zoom into the tax calculation timing.

Developer Tools - http://localhost:8080/openbravo/web/org.openbravo.retail.posterminal/?terminal=VBS-1

Elements Network Sources Timeline Profiles Resources Audits Console

3 695

Stacks Memory

Filter All Loading Scripting Rendering Painting

500 ms 1000 ms 1500 ms 2000 ms 2500 ms 3000 ms 3500 ms 4000 ms 4500 ms 5000 ms 5500 ms 6000 ms 6500 ms 7000 ms 7500 ms 8000 ms 8500 ms 9000 ms

61.6 MB – 88.4 MB

RECORDS

2510 ms 2520 ms 2530 ms 2540 ms 2550 ms 2560 ms 2570 ms 2580 ms 2590 ms 2600 ms 2610 ms 2620 ms 2630 ms 2640 ms 2650 ms 2660 ms

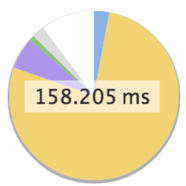
- Function Call (b1f838428...
- Function Call (b1f83842...
  - Function Call (InjectedSc...
  - Function Call (InjectedSc...
  - Function Call (InjectedSc...
  - GC Event (14.0 MB collec...
- Event (mousemove)
- Event (mousemove)
- Function Call (b1f838428...
- Function Call (b1f838428...
- Event (mousedown)
- Function Call (b1f838428...
- Function Call (b1f838428...
- Function Call (b1f838428...
- Function Call (b1f838428...

MEMORY

- Used JS Heap [65557720:7729
- Documents [1:1]

Selected Range Details

Range: 2.50 s – 2.66 s



4.609 ms Loading
122.494 ms Scripting
9.559 ms Rendering
1.577 ms Painting