Timings

Zwischendrin immer `addTiming("kleine Info was gemacht wurde")` aufrufen.

Es können auch einzelne Durchläufe einer Schleife gemessen werden: `addTiming("Info zur Schleife", "Info zum Durchlauf")`. So werden die aufeinanderfolgende Timings, bei denen der erste Parameter identisch ist, gruppiert.

Am Ende gibt `getTimings()` eine Zusammenfassung als `String` zurück. Für Schleifen wird die Gesamtdauer sowie die Anzahl der Durchläufe und die mittlere Dauer ausgegeben. Außerdem werden die Top und Flop 3 der Durchläufe (nach Dauer) angezeigt.

Zum Kopieren

```
/* Timings start */
def timings = []
def timingLast = new Date().time
def addTiming = { String title, String subTitle = null ->
  def duration = new Date().time - timingLast
  if (subTitle) {
    def value = [
      duration: duration,
      title: subTitle
    ]
    if (!timings || timings[-1].title != title) {
      timings << [
        title: title,
        values: []
      ]
    }
    timings[-1].values << value
    value.index = timings[-1].values.size()
  } else {
    timings << [
      duration: duration,
      title: title
    ]
```

```
  }
  timingLast = new Date().time
}
Closure<String> getTimings = {
  def res = new StringBuilder()
  res << "Timings:\n"
  def timingTotal = 0
  def emptyPrefix = " " * 27
  def maxDetails = 6
  timings.each { t ->
    if (t.values) {
      List vals = t.values.sort { a, b ->
        a.duration <=> b.duration ?:
          a.index <=> b.index
      }
      long sum = vals.sum { it.duration } as long
      double avg = sum / t.values.size()
      timingTotal += sum
      res << String.format("%8d ms (+%8d ms) %s (%dx ~%.2f ms)\n", timingTotal, sum, t.title,
t.values.size(), avg)
      if (vals.size() > maxDetails) vals = [*vals[0..(maxDetails / 2 - 1)], null, *vals[-
(maxDetails / 2)..-1]]
      vals.each { val ->
        res << emptyPrefix
        if (val) res << String.format("#%-4d %4dms %s\n", val.index, val.duration, val.title)
        else res << "[...]\n"
      }
    } else {
      timingTotal += t.duration
      res << String.format("%8d ms (+%8d ms) %s\n", timingTotal, t.duration, t.title)
    }
  }
  def lastDuration = new Date().time - timingLast
  timingTotal += lastDuration
  res << String.format("%8d ms total\n", timingTotal)
  return res.toString()
}
/* Timings end */
```

Beispiel

```
// do stuff
addTiming("startup")
// do more stuff
addTiming("init")
// do loop stuff 1
addTiming("fetch info", "load one...")
// do loop stuff 2
addTiming("fetch info", "load two...")
// ...
// do database stuff
addTiming("finalizing")
// do final stuff
```

Erzeugt dann z.B. mit `logI(getTimings())` sowas:

```
Timings:
       1 ms (+       1 ms) startup
      94 ms (+      93 ms) init
     753 ms (+     659 ms) fetch info (7x ~94,14 ms)
                           #4      15ms load 15F53263D95
                           #5      16ms load 15F53263DA5
                           #3      17ms load 15F53263D86
                           [...]
                           #2      20ms load 15F53263D75
                           #6      78ms load 15F53263DF3
                           #1     496ms load 15F53263D61
    2598 ms (+    1845 ms) finalizing
    2605 ms total
```