Artículo
Kurro Lopez · 4 jun, 2021 Lectura de 3 min

Visualizando la jungla de datos - Parte III. Excursión a los mapas de calor basados en %SYS.MONLBL

La cobertura del código y su optmización del rendimiento ya han surgido muchas veces, así que la mayoría de vosotors seguro que ya conocéis la herramienta SYS.MONLBL.

A menudo, un enfoque visual para revisar el código es mucho más intuitivo que los números puros. Este es principalmente el objetivo de esta serie de artículos. Esta vez vamos a hacer una pequeña excursión lejos de Python y sus herramientas, y vamos a explorar la generación de mapas de calor de los informes ^%SYS.MONLBL.

Como recordatorio rápido, un mapa de calor es solo una herramienta de visualización concreta, que nos da una visión general de los datos, en la que los colores representan un determinado valor. En nuestro caso, los datos serán líneas de código, con el tiempo que se dedica a ellas representado en colores.

^%SYS.MONLBL

Para obtener más información sobre cómo ejecutar la monitorización línea a línea, consulta esta documentación. En resumen, vamos a trabajar con un archivo CSV como salida completa de un análisis. Es mucho más útil si realmente tenemos el código fuente que estamos tratando de analizar. Asegúrate de compilar tu código con la marca k (mantén la fuente).

Cómo preparar la salida

Como salida objetivo vamos a utilizar un documento ya preparado en html. Solo incluirá un diseño muy básico y una pequeña función en javascript para realizar el coloreado final.

```
<!doctype html>
<html class="no-js" lang="">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="x-ua-compatible" content="ie=edge">
        <title></title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <!-- Place favicon.ico and apple-touch-icon.png in the root directory -->
        <link rel="apple-touch-icon" href="apple-touch-icon.png">

        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/normalize
/4.2.0/normalize.min.css">

        <!--<link rel="stylesheet" href="css/main.css"> -->
            <style>
        table, th, td {
            width:"100%";
            border: 1px solid black;
            border-collapse: collapse;
```

```
        }
        pre {
            margin:1px;
        }
        th {
            text-align: left;
        }
    </style>
```

Por motivos de seguridad, este bloque no puede estar editado como código. El siguiente bloque ha de ir entre etiqueta Script

```
    function rgba(r, g,b,a){
        r = Math.floor(r);
        g = Math.floor(g);
        b = Math.floor(b);
        return ["rgba(",r,",",g,",",b,",",a,")"].join("");
    }
    function colorize() {
        var rows=$("#data tr")
        var max=Math.max.apply(Math,rows.slice(1,rows.length).map(function(){ return
this.childNodes[2].textContent}))
        for (i=1;i<rows.length;i++){
            var val=rows[i].childNodes[2].textContent;
            var c=(Math.pow(1-val/max,3))*255;
            var col=rgba(255,c,c,0.7);
            console.log(col);
            rows[i].style.backgroundColor=col;
        }
    }
```

Aquí iria el cierre de la etiqueta script

```
    </head>

    <body onload="colorize()">
        <!--[if lt IE 8]>
            <p class="browserupgrade">You are using an <strong>outdated</strong> brow
ser. Please <a href="http://browsehappy.com/">upgrade your browser</a> to improve you
r experience.</p>
        <![endif]-->
```

Por motivo de seguriad, este bloque no puede estar editado como código

```
<script src="http://code.jquery.com/jquery-1.12.4.min.js"></script>
<script>window.jQuery || document.write('<script src="js/vendor/jquery-1.12.4.min.js"><\/script>')</script>
```

```
<table id="data">
<tr><th>Routine</th><th>Line</th><th>Total Time</th><th>Code</th></tr>
<!--output-->
</table>


    </body>
```

```
</html>
```

Analizar y reunir la información

Por medio de los siguientes scripts se obtiene la información relevante del CSV generado y se introduce en nuestra plantilla:

monlbl.sh

```
#!/bin/bash

cat $1|grep -vi totals| awk -F"," 'FNR>1 {out="<tr><td>"$1"</td>" "<td>" $2 "</td><td
>" $54 "</td><td><pre>"; for(i=55;i<=NF;i++){out=out$i","}; out=substr(out, 1, length
(out)-1) "</pre></td></tr>"; print out }'
```

gen-heatmap.sh

```
#!/bin/bash
./monlbl.sh $1 > /tmp/temp.data
sed -e '/<!--output-->/r/tmp/temp.data' template.html
```

Que llamamos de esta forma:

```
./gen-heatmap.sh /tmp/report.csv > heatmap.html
```

Resultado final

Ajustables

Si echas un vistazo más de cerca a la función de coloreado en nuestra plantilla, verás que no estoy usando un mapeo lineal para los tiempos:

```
    function colorize() {
        var rows=$("#data tr")
        var max=Math.max.apply(Math,rows.slice(1,rows.length).map(function(){ return
this.childNodes[2].textContent}))
        for (i=1;i<rows.length;i++){
            var val=rows[i].childNodes[2].textContent;
            var c=(Math.pow(1-val/max,3))*255;
            var col=rgba(255,c,c,0.7);
            console.log(col);
            rows[i].style.backgroundColor=col;
        }
    }
```

Encontré que esto funcionaba bastante bien con los ejemplos que probé, pero los resultados pueden variar. Evidentemente, se puede aumentar el exponente para llevarlo más al rojo, o viceversa.

Código

Puedes encontrar todos los archivos relevantes aquí

#Herramientas #Rendimiento #Visualización #Caché

URL de fuente:https://es.community.intersystems.com/post/visualizando-la-jungla-de-datos-parte-iii-excursi%C3%B3n-los-mapas-de-calor-basados-en-sysmonlbl