Artículo
Alberto Fuentes · 19 oct, 2021 Lectura de 5 min

Open Exchange

Analytics de Juego de Tronos o ¿Cómo de larga es la lista de Arya Stark?

¡Hola desarrolladores!

Últimamente he estado probando el módulo csvgen y buscaba un fichero CSV para probar. Resulta que encontré un fichero muy interesante en Data.World con estadísticas sobre los episodios de *Game of Thrones* (Juego de Tronos). Estadísticas sobre muertes .

¡Han documentado todos los asesinatos a lo largo de las 8 temporadas y han anotado dónde, quién, qué clan y con qué arma ha matado a otro personaje!

Así que lo he importado para hacer un cuadro de mando con IRIS Analytics.

No te preocupes, Jon. Con este cuadro de mando podremos hacer que averigües algo. Como resultado de esta «investigación» tenemos este cuadro de mando:

Algunos datos

El «nivel de sangre» crece de temporada en temporada y alcanza su máximo en la Temporada 8.

Hay 68 armas, y la más peligrosa es el fuego de dragón siendo Daenerys Targaryen la heroína más sangrienta con +1000 víctimas.

Si excluimos el fuego de dragón como arma, entonces Cersei Lannister encabeza la lista de asesinos con 199 víctimas a lo largo de las 8 temporadas.

Lo que me ha sorprendido es que la Casa Stark ha matado a más soldados Lannister que viceversa: 34 Vs. 14.

¿Cómo funciona todo esto?

Puede montarse más o menos en media hora.

En primer lugar, se puede partir de un repositorio de plantilla como objectscript-docker-template.

El módulo csvgen te da la opción de procesar un CSV y generar la clase correspondiente en IRIS para luego importar los datos, y todo en un solo comando. En nuestro caso, lo que nos ha hecho falta es lo siguiente:

```
set fn="/irisdev/app/data/game_of_thrones_deaths_collecti.csv"
set status=##class(community.csvgen).Generate(fn,",",1,,.tResults)
```

Esta línea en el código.

Como resultado, obtenemos la clase generada, donde como ves csvgen nos ha incluido los tipos de datos también:

```
Class shvarov.GOT.Deaths Extends %Library.Persistent [ Not Abstract, DdlAllowed, Not
LegacyInstanceContext, ProcedureBlock ]
{

Property name As %Library.String(MAXLEN = 250) [ SqlColumnNumber = 2 ];

Property allegiance As %Library.String(MAXLEN = 250) [ SqlColumnNumber = 3 ];

Property season As %Library.Integer(MAXVAL = 2147483647, MINVAL = -2147483648) [ SqlC
olumnNumber = 4 ];

Property episode As %Library.Integer(MAXVAL = 2147483647, MINVAL = -2147483648) [ Sql
ColumnNumber = 5 ];

Property location As %Library.String(MAXLEN = 250) [ SqlColumnNumber = 6 ];

Property killer As %Library.String(MAXLEN = 250) [ SqlColumnNumber = 7 ];

Property killershouse As %Library.String(MAXLEN = 250) [ SqlColumnNumber = 8, SqlFiel
dName = killers_house ];

Property method As %Library.String(MAXLEN = 250) [ SqlColumnNumber = 9 ];

Property deathno As %Library.Integer(MAXVAL = 2147483647, MINVAL = -2147483648) [ Sql
ColumnNumber = 10, SqlFieldName = death_no ];
```

Lo único que hay que cambiar manualmente es una modificación para añadir una comprobación después del %Save() en el método Import() generado. Sin esa modificación, nunca sabrías el motivo por el que no has logrado importar datos.

A continuación, se han creado la clase del cubo — con el *Architect* — , pivot queries — con el *Analyzer* — y un cuadro de mando — con el *IRIS Analytics Portal* —.

Finalmente se visualiza el cuadro de mando con DSW.

Se cocina todo junto en una imagen Docker utilizando este Dockerfile, donde se utiliza ZPM para instalar:

- csvgen - para generar CSV e importar datos.
- isc-dev - para exportar fácilmente artefactos de IRIS Analytics.
- dsw - para visualizar los datos.

Todo ello desarrollado en VSCode utilizando el plugin de ObjectScript para VSCode y una imagen Docker de InterSystems IRIS Community Edition 2020.2

Y además en este caso ha sido desplegado — con cada *push* — al *GCP Kubenertes Engine (GKE)* — utilizando el Workflow de GitHub Actions. En particular, este cuadro de mando se re-despliega con cada *push* que se hace en la rama *master*.

Como resultado, podéis echarle un vistazo al cuadro de mando interactivo en funcionamiento.

La calidad del código ObjectScript se evalúa continuamente utilizando ObjectScript Quality vía este archivo de workflow y puede ser examinado aquí.

¡Cualquier colaboración es bienvenida!

P.D: había 68 personas en la lista de Arya

#Analítica #CSV #Docker #ObjectScript #InterSystems IRIS
Ir a la aplicación en InterSystems Open Exchange