Artículo
Ricardo Paiva · 12 sep, 2022 Lectura de 5 min

Open Exchange

Anonimización de datos con iris-Disguise

En primer lugar, ¿qué es la anonimización de datos?

Según la Wikipedia:

> **La anonimización** es un tipo de sanitización de información cuya intención es la protección de la privacidad. Es el proceso de eliminar información personal de los conjuntos de datos, de modo que las personas que son descritas por los datos permanecen en el anonimato.

> En otras palabras, la anonimización de datos es un proceso que conserva los datos pero mantiene la fuente anónima.
> Según la técnica de anonimización que se adopte, los datos son editados, ocultados o sustituidos.

Y ese es el propósito de iris-Disguise, ofrecer un conjunto de herramientas de anonimización.

Lo puedes usar de dos formas diferentes, por método de ejecución o especificar tu estrategia de anonimización dentro de la definición de la clase persistente en sí misma.

La actual versión de iris-Disguise ofrece 6 estrategias para anonimizar datos:

- ***Destruction*** **(Destrucción)**
- ***Scramble*** **(Codificación)**
- ***Shuffling*** **(Reorganización)**
- ***Partial masking*** **(Ocultación parcial)**
- ***Randomization*** **(Distribución aleatoria)**
- ***Faking*** **(Falsificación)**

Vamos a explicar cada estrategia. Mostraré un método de ejecución con un ejemplo como he mecionado, y también mostraré cómo aplicarlo dentro de la definición de la clase persistente.
Para usar **iris-Disguise** de esta forma, necesitaréis "*llevar unas gafas de disfraz*".
En la clase persistente, se puede extender la clase **dc.Disguise.Glasses** y cambiar cualquier propiedad con los tipos de datos con la estrategia que prefieras.
Tras ello, en cualquier momento, simplemente llama al método **DisguiseProcess** en la clase. Todos los valores serán sustituidos usando la estrategia del tipo de datos.

Así que... abrochaos el cinturón que empezamos!

Destruction **(Destrucción)**

Esta estrategia reemplazará una columna entera con una palabra ('CONFIDENTIAL' (CONFIDENCIAL) por defecto).

```
Do ##class(dc.Disguise.Strategy).Destruction("classname", "propertyname", "Word to re
place")
```

El tercer parámetro es opcional. Si no se aporta, se usará la palabra 'CONFIDENTIAL' (CONFIDENCIAL).

```
Class packageSample.FictionalCharacter Extends (%Persistent, dc.Disguise.Glasses)
{
Property Name As dc.Disguise.DataTypes.String(FieldStrategy = "DESTRUCTION");
}
```

```
Do ##class(packageSample.FictionalCharacter).DisguiseProcess()
```

Scramble **(Codificación)**

Esta estrategia codificará todos los caracteres de una propiedad.

```
Do ##class(dc.Disguise.Strategy).Scramble("classname", "propertyname")
```

```
Class packageSample.FictionalCharacter Extends (%Persistent, dc.Disguise.Glasses)
{
Property Name As dc.Disguise.DataTypes.String(FieldStrategy = "SCRAMBLE");
}
```

```
Do ##class(packageSample.FictionalCharacter).DisguiseProcess()
```

Shuffling **(Reorganización)**

Shuffling reorganizará todos los valores de una propiedad dada. No es una estrategia de ocultación porque trabaja "verticalmente".
Esta estrategia es útil para relaciones, porque se mantendrá la integridad referencial.
Hasta esta versión, este método solo funciona en **relaciones de uno a muchos**.

```
Do ##class(dc.Disguise.Strategy).Shuffling("classname", "propertyname")
```

```
Class packageSample.FictionalCharacter Extends (%Persistent, dc.Disguise.Glasses)
{
Property Name As %String;
Property Weapon As dc.Disguise.DataTypes.String(FieldStrategy = "SHUFFLING");
}
```

```
Do ##class(packageSample.FictionalCharacter).DisguiseProcess()
```

Partial Masking (Ocultación parcial)

Esta estrategia ocultará parte de los datos. Por ejemplo, el número de una tarjeta de crédito puede ser sustituido por 456X XXXX XXXX X783

```
Do ##class(dc.Disguise.Strategy).PartialMasking("classname", "propertyname", prefixLe
ngth, suffixLength, "mask")
```

PrefixLength, suffixLength y mask son opcionales. Si no se aporta, se usarán los valores por defecto.

```
Class packageSample.FictionalCharacter Extends (%Persistent, dc.Disguise.Glasses)
{
Property Name As %String;
Property SSN As dc.Disguise.DataTypes.PartialMaskString(prefixLength = 2, suffixLengt
h = 2);
Property Weapon As %String;
}
```

```
Do ##class(packageSample.FictionalCharacter).DisguiseProcess()
```

Randomization **(Distribución aleatoria)**

Esta estrategia generará sencillamente datos aleatorios. Hay tres tipos de distribución aleatoria: integer, numeric y

date.

```
Do ##class(dc.Disguise.Strategy).Randomization("classname", "propertyname", "type", f
rom, to)
```

type: "integer", "numeric" o "date". "integer" es el predeterminado.

"from" y "to" son opcionales. Es para definir el rango de distribución aleatoria.
Para el tipo "integer" el rango por defecto es de 1 a 100. Para el tipo "numeric" el rango por defecto es de 1.00 a 100.00.

```
Class packageSample.FictionalCharacter Extends (%Persistent, dc.Disguise.Glasses)
{
Property Name As %String;
Property Age As dc.Disguise.DataTypes.RandomInteger(MINVAL = 10, MAXVAL = 25);
Property SSN As %String;
Property Weapon As %String;
}

Do ##class(packageSample.FictionalCharacter).DisguiseProcess()
```

Faking (Falsificación)

La idea de la falsificación es reemplazar datos con valores aleatorios pero plausibles.
iris-Disguise proporciona una pequeño conjunto de métodos para generar datos falsos.

```
Do ##class(dc.Disguise.Strategy).Fake("classname", "propertyname", "type")
```

type: "firstname", "lastname", "fullname", "company", "country", "city" y "email"

```
Class packageSample.FictionalCharacter Extends (%Persistent, dc.Disguise.Glasses)
{
Property Name As dc.Disguise.DataTypes.FakeString(FieldStrategy = "FIRSTNAME");
Property Age As %Integer;
Property SSN As %String;
Property Weapon As %String;
}

Do ##class(packageSample.FictionalCharacter).DisguiseProcess()
```

¿Qué os parece?

Me gustaría conocer vuestra opinión y leer vuestros comentarios - qué os parece, si se ajusta a vuestras necesidades y qué funcionalidades echais de menos.

Y quería agradecer de forma especial a @Henrique Dias, @Oliver Wilms, @Robert Cemper, @Yuri Marx y @Evgeny Shvarov sus comentarios, revisiones, sugerencias y valiosos aportes, que me han inspirado tanto y ayudado a crear y mejorar iris-Disguise.

#Mejores prácticas #InterSystems IRIS
Ir a la aplicación en InterSystems Open Exchange