Artigo
Henry Pereira · Dez. 1, 2021 5min de leitura

Open Exchange

Anonimização de dados, apresentando iris-Disguise

Em primeiro lugar, o que é anonimização de dados?

De acordo com a Wikipedia:

> O **anonimização de dados** é um tipo de higienização de informações cujo objetivo é a proteção da privacidade. É o processo de remoção de informações de identificação pessoal dos conjuntos de dados, para que as pessoas que os dados descrevem permaneçam anônimas.

Em outras palavras, anonimização retém os dados, mas mantém a fonte anônima.
Dependendo da técnica de anonimização adotada os dados são editados, mascarados ou substituídos.

E é este o propóstido do iris-Disguise, providenciar várias ferramentas de anonimização.

Você pode utilizar de duas maneira, executando um método ou especificando a estratégia de anonimização dentro da definição da classe persistente.

Na versão atual o iris-Disguise oferece 6 estratégias de anonimização:

- **Destruction**

- **Scramble**
- **Shuffling**
- **Partial Masking**
- **Randomization**
- **Faking**

Explicarei cada estratégia, mostrarei a execução por método e como mencionado, também mostrarei como usar pela definição da classe persistente.
Para usar o **iris-Disguise** desta maneira você precisa do "*óculos de disfarce*".
Na classe persistente, extenda da classe **dc.Disguise.Glasses** e mude o tipo das propriedades de acordo com a estratégia que escolher.
Feito isto, a qualquer momento, é só chamar o método **DisguiseProcess** na própria classe. Todos os valores serão substituidos utilizando a estratégia definida na propriedade.

Vamos lá!

Destruction

Esta estratégia substitui a coluna inteira com uma palavra ('CONFIDENTIAL' é o padrão).

```
Do ##class(dc.Disguise.Strategy).Destruction("nomeDaClasse", "nomeDaPropriedade", "pa
lavra")
```

O terceiro parâmetro é opcional. Caso não informado 'CONFIDENTIAL' será usado.

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

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

Scramble

Esta estratégia embaralha todos os caracteres da propriedade.

```
Do ##class(dc.Disguise.Strategy).Scramble("nomeDaClasse", "nomeDaPropriedade")
```

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

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

Shuffling

Shuffling vai rearranjar todos valores da propriedade. Não é considerado uma estratégia de máscara porque trabalha "verticalmente".
É uma estratégia útil para relationship pois mantém a integridade.
Até a versão atual, funciona apenas para **relacionamentos one-to-many**.

```
Do ##class(dc.Disguise.Strategy).Shuffling("nomeDaClasse", "nomeDaPropriedade")
```

```
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

Esta estratégia ofusca parte dos dados, um número de cartão de crédito, por exemplo, será substituido por 456X XXXX XXXX X783

```
Do ##class(dc.Disguise.Strategy).PartialMasking("nomeDaClasse", "nomeDaPropriedade",
prefix, suffix, "mascara")
```

Prefix, suffix e mascara são opcionais.

```
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

Esta estratégia gera dados puramente aleatórios. São 3 tipos de randomização: integer, numeric and date.

```
Do ##class(dc.Disguise.Strategy).Randomization("nomeDaClasse", "nomeDaPropriedade", "
type", from, to)
```

type: "integer", "numeric" ou "date". "integer" é o padrão.

from e **to** são opcionais. Servem para definir a faixa de randomização.
Para o tipo integer o default é de 1 à 100. Para tipo numeric o default é de 1.00 à 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()
```

Fake Data

A ideia do Faking é substituir por valores aleatórios porém plausíveis.
iris-Disguise possui uma pequena quantidade de métodos para gerar fake data.

```
Do ##class(dc.Disguise.Strategy).Fake("nomeDaClasse", "nomeDaPropriedade", "type")
```

type: "firstname", "lastname", "fullname", "company", "country", "city" and "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()
```

Quero ouvir você!

Feedbacks e ideias são muito bem vindas!

Me deixe saber o que pensa sobre esta ferramenta, como te atenderia e quais funcionalidades estão faltando.

Quero deixar um agradecimento especial para @Henrique Dias, @Oliver Wilms, @Robert Cemper, @Yuri Marx e @Evgeny Shvarov que comentaram, revisaram, sugeriram e fizeram ricas discussões que me inspiraram para criar e melhorar o iris-Disguise.

Se gostou da ideia poderia deixar o seu voto para o iris-disguise no concurso atual, obrigado.

#InterSystems IRIS

Confira o aplicativo relacionado no InterSystems Open Exchange