Criando coleções de dados em XML com Unit especializada

Explicação de como utilizar a unit para criar coleções de dados em XML

- Descrição da Unit
- Criando requests em XML para atender o padrão com coleções

Descrição da Unit

A *Unit* `utuFieldCollection` fornece um conjunto de métodos que permitem a criação de coleções de dados. Ela é descrita abaixo:

```
unit utuFieldCollection;
interface
uses
  acuFramework, acuObject, utuMessage;
type
  utFieldCollection = class    class function AddCollection(const piFieldName: string; const
poField: utField): utField;
    class procedure AddCurioNamespace(const piField: utField);    class procedure
SetToUnwrap(const piField: utField);
  end;
implementation
uses
  sysUtils, ucuUseCase;
class function utFieldCollection.AddCollection(const piFieldName: string;  const poField:
utField): utField;
var
  lField: utField;
begin
  lField := poField.AddField(piFieldName);  lField.AddAttribute('curio:collection').AsBoolean :=
true;
  Result := lField;
end;
class procedure utFieldCollection.AddCurioNamespace(const piField: utField);
begin
  piField.AddAttribute('xmlns:curio').AsString := 'http://www.evologica.com.br/';
end;
```

```
class procedure utFieldCollection.SetToUnwrap(const piField: utField);
begin
  piField.AddAttribute('curio:unwrap').AsBoolean := true;
end;
initialization
finalization
end.
```

O método `AddCollection` cria um novo campo no XML anotado como uma coleção. Ele recebe como parâmetro o nome da coleção e o campo pai da coleção.

O método `SetToUnwrap` é utilizado para demarcar quais campos do XML devem ser "desembrulhados", isto é, devem ter apenas seus valores internos considerados. Ele recebe como parâmetro o campo a ser anotado. Obs: um campo anotado com `unwrap` não deve ter filhos, apenas um valor único.

O método `AddCurioNamespace` define o *namespace* `curio` que é utilizado para definir uma coleção e se os elementos dentro de uma coleção devem ser "desembrulhados". Ele recebe como parâmetro o campo raiz, então sugere-se passar o campo do `SYSMSG` como seu argumento.

Criando requests em XML para atender o padrão com coleções

Para satisfazer os requerimentos de conversão entre XML/JSON, é necessário entender como o conversor transforma as *requests* recebidas.

XML -> JSON:

```
//Coleção pura
//Para garantir a padronização, as tags filhas devem ser do mesmo tipo.//Nas conversões de JSON para XML com Collection, os filhos sempre serão colocados com tag 'Item')
...
<Colecao curio:collection="true">   <---->   Colecao: [  <Item att1="1" att2="2" />        <----> 
    { _att1: "1", _att2: "2" },  <Item att1="3" att2="4" />        <---->     { _att1: "3", _att2: "4" },  <Item att1="5" att2="6" />        <---->     { _att1: "5", _att2: "6" }
</Colecao>                          <---->   ]
...
//Coleção com unwrap
...
<Colecao curio:collection="true" curio:unwrap="true">   <---->   Colecao: [
<Item>1</Item>                                          <---->     "1",
<Item>2</Item>                                          <---->     "2",
<Item>3</Item>                                          <---->     "3"
</Colecao>                                                <---->   ]
...
```

```
//Elemento simples com filho(s)
...
<ElementoPai>                       <---->   ElementoPai: {  <ElementoFilho1 />
<---->     ElementoFilho1: ...,  <ElementoFilho2 />               <---->     ElementoFilho2: ...
</ElementoPai>                      <---->   }
...
//Elemento simples
...<Elemento att1="1" att2="2">5</Elemento>    <---->   Elemento: { _: "5", _att1: "1", _att2:
"2" }
...
```