Questa libreria PHP consente di creare e manipolare agevolmenete una Fattura Elettronica nel formato XML richiesto dal Sistema di Interscambio (SdI) dell'Agenzia delle Entrate.

Principali Utilizzi

- Creare un file formato XML per essere esportato dal tuo gestionale e/o inoltrato ad un intermediario accreditato per la trasmissione dei documenti al SdI, quale ad esempio "https://fattura-elettronica-api.it".
- Estrapolare i dati da file importati o ottenuti dai suddetti servizi intermediari verso il SdI.

La libreria è compatibile con PHP 5.6 e PHP 7.x

Installazione

Manualmente

- Clonare o scaricare il repository
- `require_once('/path/to/FatturaElettronicaXML/src/FatturaElettronica.php');`

Nuova fattura

Fattura FPA (verso la pubblica amministrazione):

```
$invoice = new FatturaElettronica('FPA12');
```

Fattura FPR (verso soggetti privati):

```
$invoice = new FatturaElettronica('FPR12');
```

Caricamento di una fattura esistente

```
$invoice = new FatturaElettronica('/path/to/invoice.xml');
```

Oppure:

```
$invoice = new FatturaElettronica('FPR12');

$invoice->load('/path/to/invoice.xml');
```

Individuare gli elementi

In generale, per individuare gli elementi che si vogliono leggere o modificare all'interno dell'XML sono necessari:

- il nome o path dell'elemento che si vuole leggere o modificare
- (opzionale) un nodo *context* a cui appartiene l'elemento (stringa o `\DOMNode`)

La sintassi da utilizzare è quella XPath, con le seguenti facilitazioni.

Path assoluto

Se il path è assoluto (comincia con `/`), esso è in realtà relativo al nodo root del documento. Nel caso di una fattura elettronica:

```
$this->getValue('/FatturaElettronicaHeader/DatiTrasmissione/CodiceDestinatario');
```

è equivalente a una query effettuata con `\DOMXPath` per

```
/p:FatturaElettronica/FatturaElettronicaHeader/DatiTrasmissione/CodiceDestinatario
```

Un path assoluto non può avere *context*.

Path relativo

A un path relativo viene sempre anteposto `//`:
"Selects nodes in the document from the current node that match the selection no matter where they are". (XPath Syntax)

Se è presente un *context*, viene anteposto `.//`. Il punto specifica che si intende partire dal nodo *context*.
È possibile anche specificare predicati XPath, ad esempio

```
$invoice->getValue('DettaglioLinee[2]/NumeroLinea');
```

leggerà il valore di `NumeroLinea` del secondo blocco `DettaglioLinee`. La numerazione parte da 1 e non da 0, come da sintassi XPath.

Tag

Se viene fornito solo il nome dell'elemento (non sono presenti `/`), vengono seguite le stesse regole dei path relativi. Ad esempio per selezionare `ModalitaPagamento`:

```
$invoice->getValue('ModalitaPagamento');
```

Con un *context*:

```
$invoice->getValue('NumItem', 'DatiContratto');
```

Leggere/modificare i valori

getValue(string $expr, $context = null)

Restituisce il valore dell'elemento individuato da `$expr/$context`, se `$expr/$context` restituisce un elemento univoco, altrimenti genera un'eccezione.
Esempi validi:

```
$invoice->getValue('ProgressivoInvio');

$invoice->getValue('Sede/Indirizzo', 'CedentePrestatore');
```

Esempi non validi:

```
$invoice->getValue('IdPaese');

$invoice->getValue('Sede/Indirizzo', 'FatturaElettronicaHeader');
```

Come detto, `$context` può essere un oggetto `\DOMNode`.

setValue(string $expr, $value, $context = null)

Setta il valore `$value` per l'elemento individuato da `$expr/$context`, se `$expr/$context` restituisce un elemento univoco, altrimenti genera un'eccezione.

setValueToAll(string $expr, $value, $context = null)

Setta il valore `$value` per tutti gli elementi individuati da `$expr/$context`.

setValues($context, array $array)

Per ogni coppia chiave/valore `$k/$v` dell'array, setta il valore `$v` per l'elemento individuato da `$k/$context`, se `$k/$context`restituisce un elemento univoco, altrimenti genera un'eccezione.
Esempio:

```
$invoice->setValues('CessionarioCommittente', array(

    'CodiceFiscale' => '02313821007',

    'Anagrafica/Denominazione' => 'AMMINISTRAZIONE',

));
```

setValuesToAll($context, array $array)

Per ogni coppia chiave/valore `$k/$v` dell'array, setta il valore `$v` per tutti gli elementi individuati da `$k/$context`.

setValuesFromArray($context, array $array)

Dato un array che rispecchia fedelmente una porzione di struttura XML di un certo nodo `$context`, setta ricorsivamente i rispettivi valori.
Esempio:

```
$array =  array(

    'DatiAnagraficiVettore' => array(

        'IdFiscaleIVA' => array(

            'IdPaese' => 'IT',

            'IdCodice' => '09876543210'

        ),

        'Anagrafica' => array(

            'Denominazione' => 'TRASPORTO SRLS'

        ),

        'NumeroLicenzaGuida' => 'AA090909'

    ),

    'MezzoTrasporto' => 'Mezzo',

    'CausaleTrasporto' => 'La causale del traporto',

    'NumeroColli' => '1',

    'Descrizione' => 'La descrizione'

);
```

```
$invoice->setValuesFromArray('DatiTrasporto', $array);
```

Lotto di fatture

addBody(int $n = 1)

Aggiunge n `FatturaElettronicaBody` alla fattura.
Poiché clona il primo body, verranno clonati anche tutti gli eventuali valori al suo interno.

getBody(int $bodyIndex = 1)

Restituisce l'oggetto `\DOMNode` relativo all'i-esimo body.
Come da sintassi XPath, l'indice parte da 1.

setBodyCount(int $n)

Setta il numero complessivo di `FatturaElettronicaBody`.
Può essere usato alternativamente a `addBody`.
Ad esempio per creare un lotto di 5 fatture:

```
$invoice->setBodyCount(5);

// oppure

$invoice->addBody(4);
```

Più linee di dettaglio

addLineItem(int $n, int $bodyIndex = 1)

Aggiunge `$n` linee di dettaglio al body `$bodyIndex`.
Ad esempio per aggiungere 3 linee di dettaglio al secondo body:

```
$invoice->addLineItem(3, 2);
```

getLineItem(int $i, int $bodyIndex = 1)

Restituisce l'oggetto `\DOMNode` relativo alla i-esima linea di dettaglio del body `$bodyIndex`.

setLineItemCount(int $n, int $bodyIndex = 1)

Setta il numero complessivo di `DettaglioLinee` del body `$bodyIndex`.
Può essere usato alternativamente a `addLineItem`.

Gestione degli elementi XML

Normalmente non c'è bisogno di utilizzare i seguenti metodi per prendere/aggiungere/rimuovere elementi XML come nodi `\DOMNode` perché:

- per leggere/settare i valori degli elementi si possono usare i loro nomi o i path, come stringhe
- per aggiungere elementi `FatturaElettronicaBody` (lotto di fatture) si possono usare i metodi `addBody`, `getBody` e `setBodyCount`
- per aggiungere linee di dettaglio si possono usare i metodi `addLineItem`, `getLineItem` e `setLineItemCount`

In ogni caso a volte può servire aggiungere, rimuovere o prendere un determinato elemento della struttura XML. È possibile farlo con i seguenti metodi.

addElement($element, $parent, $beforeRef = null)

Aggiunge un elemento `$element` all'interno del nodo genitore `$parent`. Viene posizionato prima del nodo fratello `$beforeRef`, se `$beforeRef` non è nullo.

addElementsFromArray($parent, array $array)

Dato un array che rispecchia fedelmente una porzione di struttura XML di un certo nodo `parent`, aggiunge ricorsivamente i vari elementi dell'array. Il metodo viene usato nel `constructor` per creare l'intera struttura della fattura (e della notifica).

getElement($expr, $context = null)

Prende un determinato elemento individuato da `$expr/$context`, se `$expr/$context` restituisce un elemento univoco, altrimenti genera un'eccezione.

removeElement($element)

Rimuove l'elemento `$element`, se univoco. Altrimenti genera un'eccezione.

setElementCount($expr, int $n, $context = null)

Imposta la cardinalità di un certo elemento. Ad esempio per avere 3 `DatiRiepilogo`:

```
$invoice->setElementCount('DatiRiepilogo', 3);
```

O nel caso di più bodies:

```
$body2 = $invoice->getBody(2);

$invoice->setElementCount('DatiRiepilogo', 2, $body2);
```

Settaggio valori:

```
$invoice->setValue('DatiRiepilogo[1]/AliquotaIVA', '22.00', $body2);
```

```
$invoice->setValue('DatiRiepilogo[2]/AliquotaIVA', '10.00', $body2);
```

Altri metodi comuni a fatture e notifiche

`getDOM()`

Restituisce l'oggetto `\DOMDocument` relativo alla fattura.

`normalize()`

Rimuove gli elementi XML vuoti.

Utilizzato in `asXML()` e `save()`, se non esplicitamente disabilitato.
In `FatturaElettronica` divide l'elemento `DatiGeneraliDocumento/Causale` in più elementi se il valore supera i 200 caratteri (chunks da 200 caratteri ciascuno).

`query( string $expr, $context = null, bool $registerNodeNS = true )`

Restituisce una lista di elementi `\DOMNodeList` derivante dalla query `$expr/$context`.

Visualizzazione dell'XML

Per recuperare la fattura come stringa XML:

```
$invoice->asXML();
```

Di default gli elementi vuoti vengono eliminati prima di stampare la stringa XML (normalizzazione). Per visualizzarli:

```
$invoice->asXML(false);
```

Salvataggio su file

Fattura

Quando si salva una fattura, di default il nome del file è dato da:

```
IdPaese . IdCodice . _ . ProgressivoInvio . .xml
```

È possibile in ogni caso assegnare un nome arbitrario alla fattura:

```
$invoice->setFilename('filename');
```

Per settare una directory di destinazione dove salvare la fattura:

```
$invoice->setPrefixPath('path/to/dir');
```

Per settare una directory di destinazione per tutti gli oggetti `FatturaElettronica` che verranno creati:

```
FatturaElettronica::setDefaultPrefixPath('path/to/dir');
```

Per salvare la fattura:

```
$invoice->save();
```

Se è già presente un file con lo stesso nome viene generata un'eccezione. Per sovrascrivere il file:

```
$overwrite = true;

$invoice->save($overwrite);
```

Prima di salvare il file, l'XML viene normalizzato (vengono rimossi gli elementi vuoti). Per disabilitare la normalizzazione:

```
$overwrite = false;

$normalize = false;

$invoice->save($overwrite, $normalize);
```

Getters:

```
FatturaElettronica::getDefaultPrefixPath();

$invoice->getFilename();

$invoice->getPrefixPath();
```

Crediti e licenza

Questa libreria si basa sul lavoro di: Taocomp s.r.l.s. https://taocomp.com repository https://github.com/taocomp/php-e-invoice-it rilasciato con licenza GPL v3.0

L'obiettivo di questa libreria è di offrire un'alternativa più mirata per sviluppatori con specifiche esigenze operative relative all'interazione con i servizi di trasmissione/ricezione

al/dal SdI. Inoltre è stata ripristinata la compatibilità con PHP 5.6 e sono state aggiunte alcune migliorie rilevatesi necessarie sul campo.