PaxListConverter

EU-Maritime

August 19, 2016

1 Dedication

This work is dedicated to friends and foes, students and teachers, bosses and clients who have triggered my curiosity during many years. They were the true motor of this work, they provided the motivation for the hard work needed to produce this work. I did enjoy the challenge and ask them to find here the reward they have merited. Thank you folks!

2 The purpose

Some obstacles can't be jumped and the best way to handle those is to walk around.

Trying to find a unique format and hoping the rest of the world to embrace it looks like an utopia to me. I think the world uses different ways to communicate, in this particular case, the list of passengers, crew and stowaway aboard a ship to the port of call (=destination). What I will explain here is not only usable to the maritime world, it's true for all situation where a single format can't be forced.

The convention (here, in this sample application) is that the data needed is a sort of table where the columns hold the different data fields and where the lines holds data of a single person. The very first line of this table hold the title of the column. The order of the columns is not (or should'nt be) irrelevant, nor the order of the lines. Mixing the passengers and the crew doesn't matter, and if it does the it is the Filters responsability. Quite standard I believe. Here is a small list of my passengers and crew.

PaxList

CPS	FirstName	LastName	BirthDate
S	Harry	Potter	1999-12-31
P	Gildor	Inglorion	1998-11-30
C	Grima	Wormthongue	1997-10-29
C	Barliman	Butterbur	1996-09-28
P	Bilbo	Baggins	1995-08-27

There is one little thing that hurts me. Why should the birthdate be written in the YYYY-MM-DD format and not my preferred format (or just because a have the birthdates in another format), like MM/DD/YY? Well I can, like this, where date columns names are followed by an underscore and the format.

PaxList

CPS	FirstName	LastName	BirthDate_MM/DD/YY
S	Harry	Potter	12/31/99
P	Gildor	Inglorion	11/30/98
C	Grima	Wormthongue	10/29/97
C	Barliman	Butterbur	09/28/96
P	Bilbo	Baggins	08/27/95

All the formats I support can be found in the class variable **$dateFormat** of **PassengersFilter**.

3 The Design Pattern

Don't reinvent the wheel.

I was largely inspired by **Mattias Noback**'s book **Principes of Package Design** publish by **LeanPub** and his excellent talk in the PHP devroom at FOSDEM 2015. You will find a lot of similarities between his book and the code here. The most important think is to apply an existent pattern, not to find new one's. So here is the implementation I made, three times: once for the `Decoder`, once for the `Filter` and finally for the `Encoder`. I will use the `Decoder` to read a file containing the *PassengerList* from different file formats, I will use the `Filter` to verify, adapt, move, ... the content and finally I will output the filtered content using an `Encoder`.

For example I will use a `ExcelDecoder` to read a Excel sheet, compute the dates with the `Filter` and I will output the content in HTML using the `HtmlEncoder`, in Json using the `JsonDecoder` and in Xml using the `XmlDecoder`. Of course, in reality, you, probably, want to output in only one format.

The files needed to implement the Decoder part are:

- DecoderInterface (see listing 3.1.1)
- DecoderInterfaceFactory (see listing 3.1.2)
- DecoderFactory (see listing 3.1.3)
- GenericDecoder (see listing 3.1.4)

and your implementation should contain:

- CsvDecoder (to decode a comma separated file, see listing 3.1.5)

and repeat this for `Filter` and for `Encoder` (if you need them).

3.1 Decoder

3.1.1 Interface

Typically the `DecoderInterface` contains:

```php
<?php
interface DecoderInterface
{
   /**
    * @param string $format
    * @return array of dict $data
    */
    public function decode(/* string */$format);
}
```

and will be used by the specific `Decoder` (here the `CsvDecoder`).

3.1.2 FactoryInterface

Typically the `DecoderFactoryInterface` contains:

```php
<?php
interface DecoderFactoryInterface
{
    /**
     * Create a decoder for the given format
     *
     * @param string $format
     * @return DecoderInterface concrete Class defined by $format
     */
    public function createForFormat(/* string */$format);
}
```

in this way we can satisfy to the SOLID principle : 'open for extension, closed for modification'.

3.1.3 Factory

```php
<?php
require_once 'Decoder/DecoderFactoryInterface.php';

class DecoderFactory implements DecoderFactoryInterface
{
    private $factories = [];

    /**
     * Register a callable that returns an instance of
         DecoderInterface for the given format
     * @param string $format
     * @param callable $factory
     */
    public function addDecoderFactory(/* string */$format, callable
        $factory)
    {
        $this->factories[$format] = $factory;
    }

    /**
     * @param string $format
     * @return DecoderInterface concrete Class defined by $format
     */
    public function createForFormat (/* string */$format)
    {
        $factory = $this->factories[$format];

        return $factory();
    }
}
```

3.1.4 Generic

The `genericDecoder` is used to call any decoder passed as a parameter in the constructor

```php
<?php
class GenericDecoder
{
    private $decoderFactory;

    /**
     * GenericDecoder constructor.
     * @param DecoderFactory $decoderFactory
     */
    public function __construct(/*DecoderFactory*/ $params)
    {
        $this->decoderFactory = $params;
    }

    /**
     * @param array $data
     * @param string $format
     * @return mixed
     */
    public function decodeToFormat (array $data, /* string */$format)
    {
        $decoder = $this->decoderFactory->createForFormat($format);
        return = $decoder->decode($data);
    }
}
```

3.1.5 A concrete implementation : the CsvDecoder

One of the simplest decoder to program.

```php
<?php
require_once LIBRARIES.'Decoder/DecoderInterface.php';

class CsvDecoder implements DecoderInterface
{
   /**
    * @param string $data
    * @return array of dict
    */
   public function decode(/* string */$dataFile)
   {
      $dataLine = [];
      $handle = fopen($dataFile, 'rt');
      if ($handle) {
         //read first line
         $keys = fgetcsv($handle);
         //read data lines
         while ($nextLine = fgetcsv($handle)) {
            $dataLine[] = array_combine($keys, $nextLine);
         }
      }
      return $dataLine;
   }
}
```

3.2 Filter

Here a my `PassengersFilter`, it could help to understand in which conditions one could use `Filter` to improve the system, without changing the code of the core (which is close for modifications).

```php
<?php
class PassengersFilter implements FilterInterface
{
   public $fields;
   public $dateFormats;

   public function __construct()
   {
      $this->fields = [
          'CPS'         => 'CPS',  // for C rew, P ax, S towaways
          'FIRSTNAME'    => 'CP',
          'LASTNAME'     => 'CP',
            'NATIONALITY'  => 'CP',
            'RANKORRATING' => 'C',
            'TYPEOFID'     => 'CP',
            'SERIALNRID'   => 'CP',
            'SERIALNRVISA' => 'P',   //necessary ev. empty
            'EXPDATE_'     => 'CP',
            'BIRTHDATE_'   => 'CP',
            'PLACEOFBIRTH' => 'CP',
            'EMBARKATION'  => 'P',
            'DISEMBARKATION'=> 'P',
        ];
        $this->dateFormats = [
           'YYYY-MM-DD' => 'Y-m-d', 'YY-MM-DD' => 'y-m-d',
           'YYYY/MM/DD' => 'Y/m/d', 'YY/MM/DD' => 'y/m/d',
           'DD-MM-YYYY' => 'd-m-Y', 'DD-MM-YY' => 'd-m-y',
           'DD/MM/YYYY' => 'd/m/Y', 'DD/MM/YY' => 'd/m/y',
           'MM-DD-YYYY' => 'm-d-Y', 'MM-DD-YY' => 'm-d-y',
           'MM/DD/YYYY' => 'm/d/Y', 'MM/DD/YY' => 'm/d/y',
           'EXCEL'     => 'EXCEL',
        ];
    }

    /**
     * @param array of dict $data
     * @return array
     */
    public function filter (array $data)
    {
        //read first line
        $firstLine = $data[0];
        //check if the required fields are present
        $foundFormat = [];
        $missingPFields = $this->findMissingFields('P', $firstLine,
            $foundFormat);
        $missingCFields = $this->findMissingFields('C', $firstLine,
            $foundFormat);
        print_r($foundFormat);
        echo '<br/>missing fields for P ';
        print_r($missingPFields);
        echo '<br/>missing fields for C ';
        print_r($missingCFields);

        $filteredC = $this->prepareData('C', $data);
        $filteredP = $this->prepareData('P', $data);

        return array_merge($filteredC, $filteredP);
    }

    /**
```

```
     * @param string $cat
     * @param array of dict $data
     * @return array
     */
    private function prepareData(/* string */$cat, array $data)
    {
       $dataOut = [];
       foreach ($data as $row){
          if ($row['CPS'] != $cat)
             continue;

          $rowOut = [];
          foreach($row as $key => $val)
          {
             $key = strtoupper($key);
             $pos = strpos($key, '_');
             if ($pos !== false){
                $fmt = substr($key, $pos + 1);
                $fmt = $this->dateFormats[$fmt];
                $key = substr($key, 0, $pos);
             }
             switch ($key){
                case 'EXPDATE':
                case 'BIRTHDATE':
                   if ($fmt == 'EXCEL'){
                      $rowOut[$key] = $val;
                   } else {
                      $date = DateTime::createFromFormat($fmt, $val);
                      $rowOut[$key] = $date->format('Y-m-d');
                   }
                break;
                default:
                   $rowOut[$key] = strtoupper($val);
             }
          }
          $dataOut[] = $rowOut;
       }
       return $dataOut;
    }

    /**
     * @param string $cat
     * @return array
     */
    private function getFieldsFor(/* string */$cat)
    {
       $fields = [];
       foreach ($this->fields as $k => $v){
          if (strpos($v, $cat) !== false){
```

```
109             $fields[] = $k;
110          }
111       }
112 
113       return $fields;
114    }
115 
116    /**
117     * @param string $cat
118     * @param string $firstLine
119     * @param &array $foundFormat
120     * @return array
121     */
122    private function findMissingFields($cat, $firstLine,
        &$foundFormat)
123    {
124       $mandatory = $this->getFieldsFor($cat);
125       $keys = array_keys($firstLine);
126       foreach($keys as &$k){
127          $k = strtoupper($k);
128          $pos = strpos($k, '_');
129          if ($pos !== false){
130             $dateFormat = substr($k, $pos+1);
131             $k = substr($k, 0, $pos+1);
132             $foundFormat[$k] = $this->dateFormats[$dateFormat];
133          }
134       }
135       asort($keys);
136       asort($mandatory);
137       $result = array_diff($mandatory, $keys);
138       return $result;
139    }
140 }
```

I must admit that the lines 46 to 51 are not at the right place and should be moved out of the class and put in a class with output responsibilities (I still have some work to do).

3.3 Encoder

I want to show the JsonEncoder,

```
<?php
class JsonEncoder implements EncoderInterface
{
   /**
    * @param array of dict $data
```

```
     * @return string
     */
    public function encode(array $data)
    {
        $data = json_encode($data);
        $data = str_replace('\"', '', $data);

        return $data;
    }
}
```

PHP is so kind to do nearly the whole job ;-)

4 Putting everything together

The `Load` class is the main thing, it is a `CI_Controller` class from CodeIgniter 3.0.6. Since it takes over the 200 lines of code, I want to highlight only some part of it. The complete code is available, you'll have it all.

4.1 `Load`

To do

4.2 Installation

How to call the stuff from your browser.

- Install PHP (>= 5.4).
- Install CodeIgniter (>= 3.0.6).
- Run composer to get the package `phpoffice/phpexcel`.
- Install all the code included here.
- if you don't have a HTTP server type on the command line:

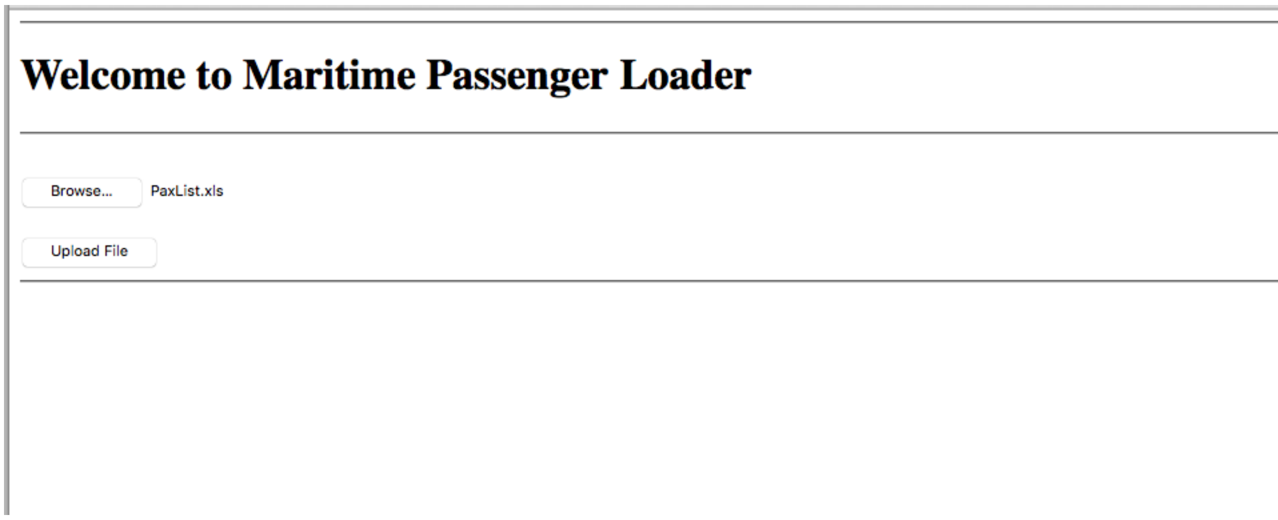
  `php -S localhost:8080 path_to_index.php`
- in your browser : localhost:8080/PaxListConverter/Load.
- you should see something like the figure 5.1.
- Choose a file from the folder `doc`.
- Click the button `Upload`.

- you should see something like the figure 5.2.
- enjoy.

5 Figures

5.1 Figure before upload

The figure when you `Browse` to PaxList in Excel and before you click `Upload File`.

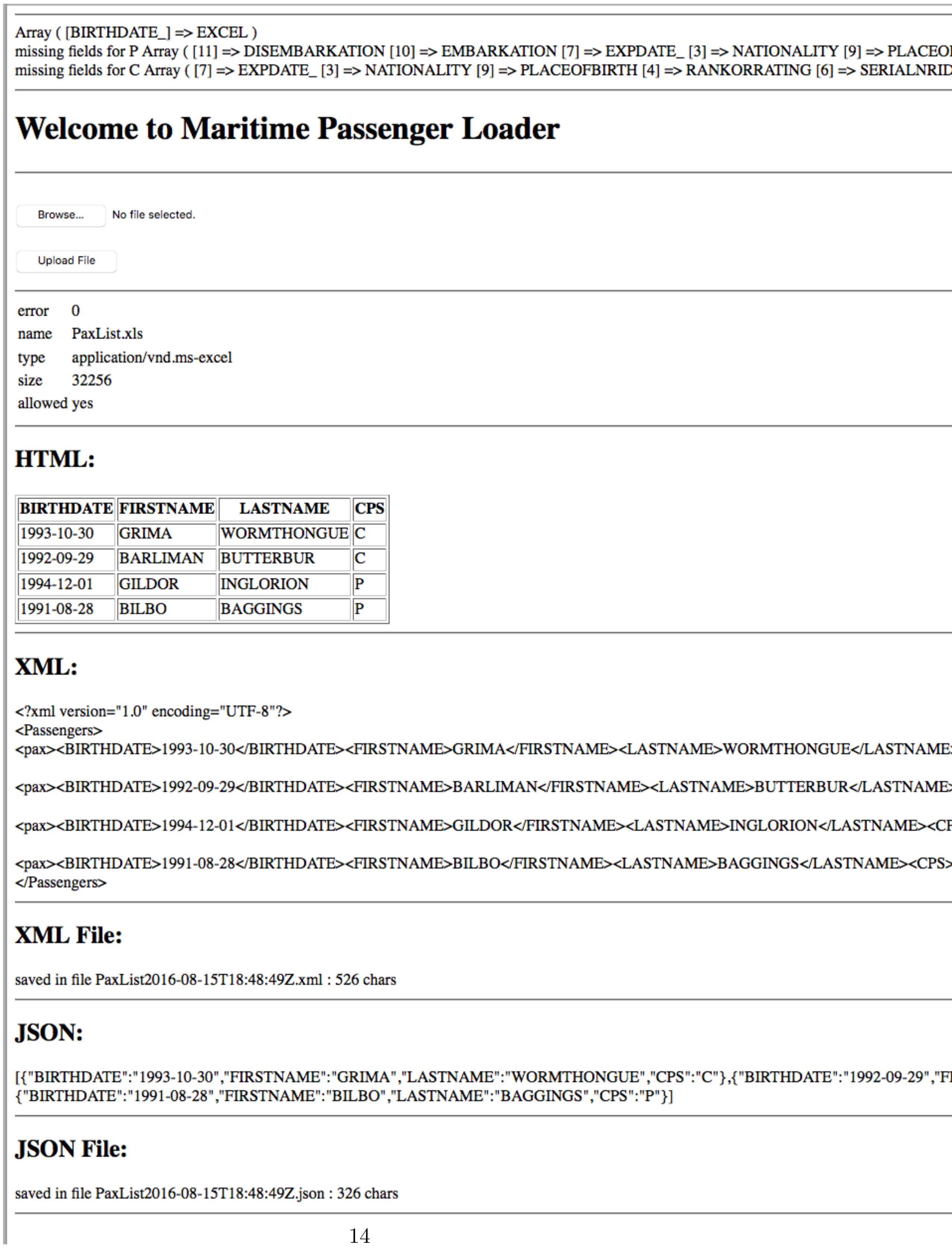

Welcome to Maritime Passenger Loader

5.2 Figure after upload

Once you click `Upload File` you get this figure with:

- A line with information about the date formats.
- A line with missing mandatory columns for Passengers.
- A line with missing mandatory columns for Crew.
- The title 'Welcome to Maritime Passenger Loader'.
- The section with `Browse` and `Upload File` buttons.
- A section about the file being read containing error, name, type, size and allowed.
- A Section HTML, being the result of the `HtmlEncoder`.
- A section XML, being the html version of the result of `XmlEncoder`.

- A line telling you the name of the xml file create by `XmlEncoder`.
- The same for JSON, in html and as a file.



```
Array ( [BIRTHDATE_] => EXCEL )
missing fields for P Array ( [11] => DISEMBARKATION [10] => EMBARKATION [7] => EXPDATE_ [3] => NATIONALITY [9] => PLACEO
missing fields for C Array ( [7] => EXPDATE_ [3] => NATIONALITY [9] => PLACEOFBIRTH [4] => RANKORRATING [6] => SERIALNRID
```

Welcome to Maritime Passenger Loader

Browse... No file selected.

Upload File

error	0
name	PaxList.xls
type	application/vnd.ms-excel
size	32256
allowed	yes

HTML:

BIRTHDATE	FIRSTNAME	LASTNAME	CPS
1993-10-30	GRIMA	WORMTHONGUE	C
1992-09-29	BARLIMAN	BUTTERBUR	C
1994-12-01	GILDOR	INGLORION	P
1991-08-28	BILBO	BAGGINGS	P

XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<Passengers>
<pax><BIRTHDATE>1993-10-30</BIRTHDATE><FIRSTNAME>GRIMA</FIRSTNAME><LASTNAME>WORMTHONGUE</LASTNAME

<pax><BIRTHDATE>1992-09-29</BIRTHDATE><FIRSTNAME>BARLIMAN</FIRSTNAME><LASTNAME>BUTTERBUR</LASTNAME

<pax><BIRTHDATE>1994-12-01</BIRTHDATE><FIRSTNAME>GILDOR</FIRSTNAME><LASTNAME>INGLORION</LASTNAME><CP

<pax><BIRTHDATE>1991-08-28</BIRTHDATE><FIRSTNAME>BILBO</FIRSTNAME><LASTNAME>BAGGINGS</LASTNAME><CPS>
</Passengers>
```

XML File:

saved in file PaxList2016-08-15T18:48:49Z.xml : 526 chars

JSON:

```
[{"BIRTHDATE":"1993-10-30","FIRSTNAME":"GRIMA","LASTNAME":"WORMTHONGUE","CPS":"C"},{"BIRTHDATE":"1992-09-29","F
{"BIRTHDATE":"1991-08-28","FIRSTNAME":"BILBO","LASTNAME":"BAGGINGS","CPS":"P"}]
```

JSON File:

saved in file PaxList2016-08-15T18:48:49Z.json : 326 chars

6 The Future

This is much more than an invitation to collaborate. This is only a starter. There is much more to do, and we will do it altogether. We are looking, not only, for more Decoder, Filters and Encoder: we are also looking for people able to translate to other programming languages like Java, C++, Python, Objective-C, ... Let us publish our work, your work in order to avoid to reinvent the wheel.

Make Europe a better place to live, make the Earth a better place to live.