# Módulo 4 – Tópicos especiais em OO

Neste workbook do módulo 4, revisaremos os principais conceitos vistos nas videoaulas.

## Tratamento de exceções

A maneira mais eficiente de lidar com situações de erro. O método que pode gerar alguma situação de inconsistência emite uma exceção (throw), que é tratada externamente.

```
<?php
require_once 'classes/CSVParser.php';

$csv = new CSVParser('clientes.csv', ';');
try {
   $csv->parse(); // método que pode lançar exceção
   while ($row = $csv->fetch()) {
      print $row['Cliente'] . " - ";
      print $row['Cidade'] . "<br>\n";
   }
}
catch (Exception $e) {
   print $e->getMessage();
}
```

## Método __get()

Intercepta uma chamada a uma propriedade inacessível. Sempre que você acessar, por exemplo, uma propriedade private do contexto externo à classe, ou uma propriedade inexistente, a requisição passará por este método.

```
<?php
class Titulo {
   private $dt_vencimento, $valor, $juros, $multa;

   public function __get($propriedade) {
      if ($propriedade == 'valor') {...}
   }
}
```

## Método __set()

Intercepta uma atribuição a uma propriedade inacessível. Sempre que, por exemplo, você tentar alterar uma propriedade private do contexto externo à classe, ou uma propriedade inexistente, passará por esse método, que receberá o nome e o valor a ser atribuído.

```
<?php
class Titulo {
   private $dt_vencimento, $valor, $juros, $multa;

   public function __set($propriedade, $valor) {
      print "Tentou gravar $propriedade = $valor. Use setValor()<br>\n";
   }
```

## Método __toString()

Retorna uma representação do objeto em string. Executado quando o objeto for exibido (Ex: print), ou concatenado em uma outra string. Ex: print “string” . $objeto;

```
<?php
class Titulo {
   private $data;

   public function __toString() {
      return json_encode($this->data);
   }
}
```

## Método __call()

Intercepta a chamada a um método inacessível. Ex: Você está executando um método que não existe nesta classe.

call.php

```
<?php
class Titulo {
   public $codigo, $dt_vencimento, $valor, $juros, $multa;

   public function __call($method, $values) {
      print "Você executou o método {$method}, com os parâmetros: ".implode(',', $values) . "<br>\n";
   }
}
```

## Simple XML

Use a biblioteca Simple XML para ler arquivos XML. Neste exemplo, estamos lendo alguns atributos de um arquivo XML.

```
<?php
// interpreta o documento XML
$xml = simplexml_load_file('paises2.xml');

echo 'Nome : '     . $xml->nome      . "<br>\n";
echo 'Idioma : '   . $xml->idioma    . "<br>\n";

echo "<br>\n";
echo "*** Informações Geográficas ***<br>\n";
echo 'Clima : '    . $xml->geografia->clima . "<br>\n";
echo 'Costa : '    . $xml->geografia->costa . "<br>\n";
```

## SPL File

Use a SPL para ler informações do sistema de arquivos. Neste exemplo, estamos lendo informações sobre um arquivo, e escrevendo outro.

```
<?php
$file = new SplFileObject('spl_file_object.php');
print 'Nome: ' . $file->getFileName() . '<br>' . PHP_EOL;
print 'Extensão: ' . $file->getExtension() . '<br>' . PHP_EOL;

$file2 = new SplFileObject("novo.txt", "w");
$bytes = $file2->fwrite('Olá Mundo PHP' . PHP_EOL);
print 'Bytes escritos ' . $bytes . PHP_EOL;
```

## Reflection

Use Reflexão para saber mais sobre classes, métodos, propriedades. Neste exemplo, estamos descobrindo sobre métodos, propriedades, e classe pai de uma classe.

```
<?php
require_once 'veiculo.php';
$rc = new ReflectionClass('Automovel');

print_r($rc->getMethods());
print_r($rc->getProperties());
print_r($rc->getParentClass());
```

## Traits

Use traits para absorver traços de comportamento em classes, sem precisar da herança. Neste exemplo, escrevemos um trait com 2 métodos, e em seguida ele é absorvido por outra classe.

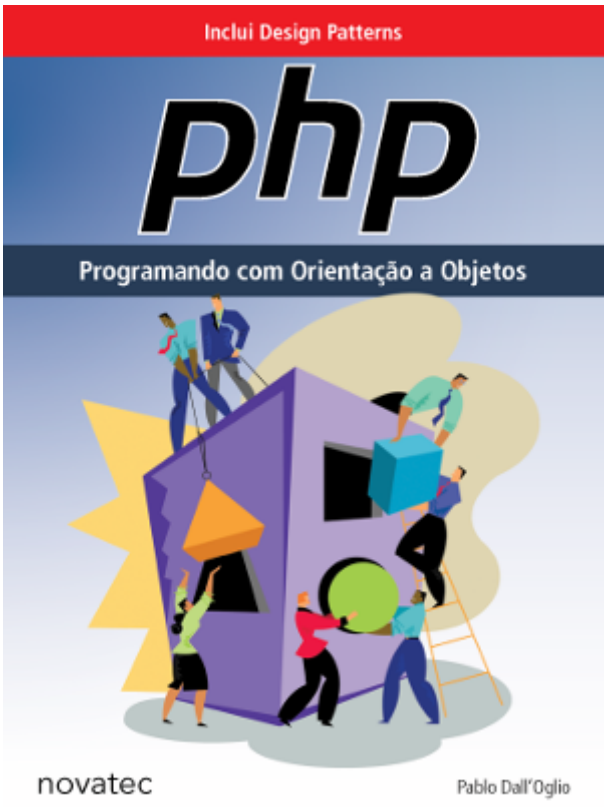
```
<?php
trait ObjectConversionTrait {
   public function toXML() {
      $data = array_flip($this->data);
      $xml = new SimpleXMLElement('<'.get_class($this).'/>');
      array_walk_recursive($data, array ($xml, 'addChild'));
      return $xml->asXML();
   }
   public function toJSON() {
      return json_encode($this->data);
   }
}
class Pessoa extends Record {
   const TABLENAME = 'pessoas';
   use ObjectConversionTrait;
}
```

## PSR

A PSR (PHP Standard Recommendation) é uma recomendação de padrões de escrita de código em PHP. Exemplos:

- Usar Namespaces para estruturar e isolar as classes.
- Os Namespaces devem iniciar com a identificação do fornecedor da classe em sua estrutura (ex.: \<Fornecedor>\<Subnível>\<Classe>).
- Os Namespaces podem ter vários subníveis.
- Uma parte inicial do Namespace corresponderá ao diretório-base de onde as classes serão carregadas. Subníveis do Namespace corresponderão a subdiretórios.
- Para que a classe seja carregada apropriadamente, o arquivo em que ela é salva deve ter exatamente o mesmo nome que a classe, acrescido do sufixo ".php". Ex.: classe \Livro\Widgets\Form\Field => *Lib/Livro/Widgets/Form/Field.php*.
- Os arquivos devem ser armazenados no formato UTF-8.
- As classes devem ser representadas no formato StudlyCaps (ex.: FormField).
- Os métodos devem ser representados no formato camelCase (ex.: getData()).
- Deve existir pelo menos uma linha em branco antes da declaração do Namespace e também antes da declaração de uso (use).
- Namespaces e classes devem seguir um autoloader-padrão.

Livro PHP Programando com Orientação a Objetos




www.adianti.com.br/phpoo