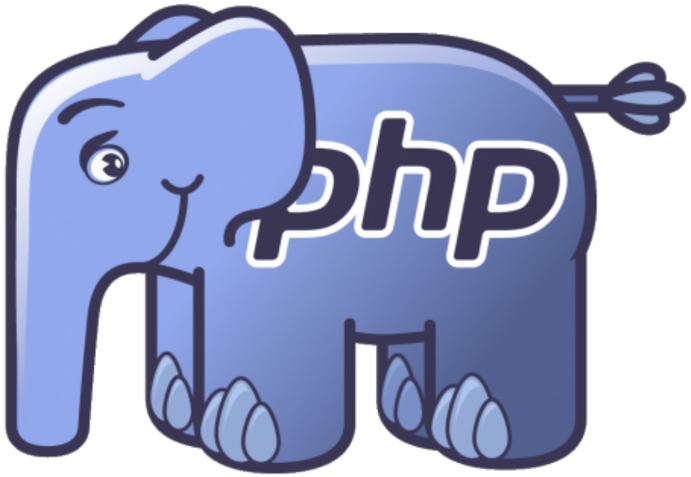

Piscine PHP

Day_06

Une interface

- **Une interface** est conçu pour des **classes** ayant des **méthodes communes**

- **Une classe héritant** d'une **interface** doit **intégrer toutes méthodes** définis dans celle-ci

Exemple - Interface

```php
<?php
interface iCharacter
{
    public function getName();
    public function setName($name);
}

class Character implements iCharacter
{
    private $name;

    public function getName()
    {
        return $this->name;
    }
    public function setName($name)
    {
        $this->name;
    }
}
```

Exemple - Interface

```php
<?php
interface iCharacter
{
    public function getName();
    public function setName($name);
}
interface iMage extends iCharacter
{
    public function attack();
    public function heal();
}
```

```php
class Mage implements iMage
{
    private $name;

    public function getName()
    {
        return $this->name;
    }
    public function setName($name)
    {
        $this->name;
    }
    public function attack()
    {
        echo "AYAAAAAAA\n";
    }
    public function heal()
    {
        echo "I'm healing\n";
    }
}
```

Des classes abstraites ?

- Création **d’interface** avec des **définitions**
- **Optimiser l’héritage** en ne dupliquant pas certaines fonctions

Exemple - Classe Abstraite

```php
<?php
abstract class Character
{
    abstract protected function say();
    abstract protected function getType();

    public function printType()
    {
        echo $this->getTyped()."\n";
    }
}

class Mage extends Character
{
    protected function say()
    {
        echo "Piou piou piou\n";
    }

    protected function getType()
    {
        return "Mage";
    }
}
```