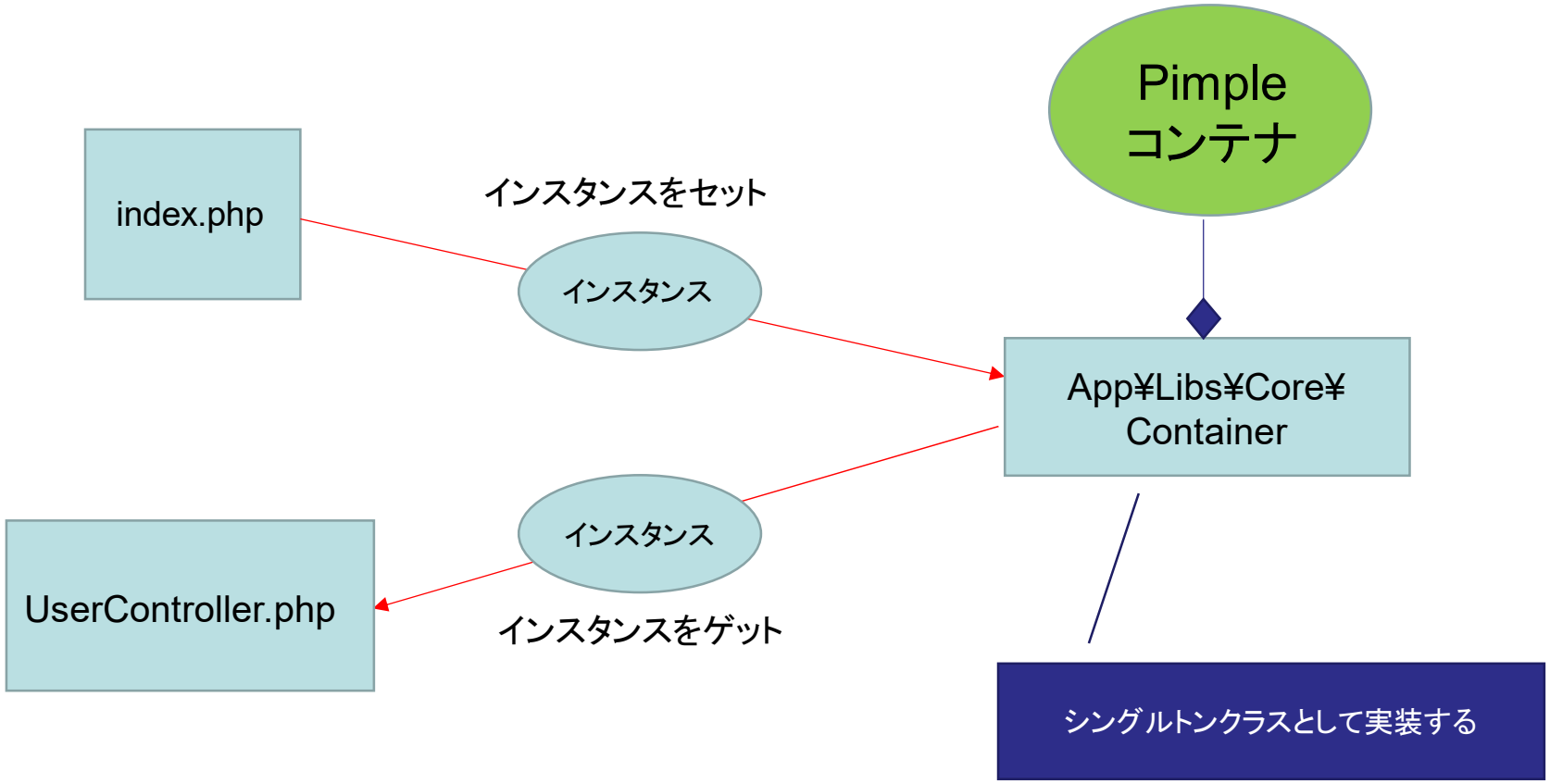
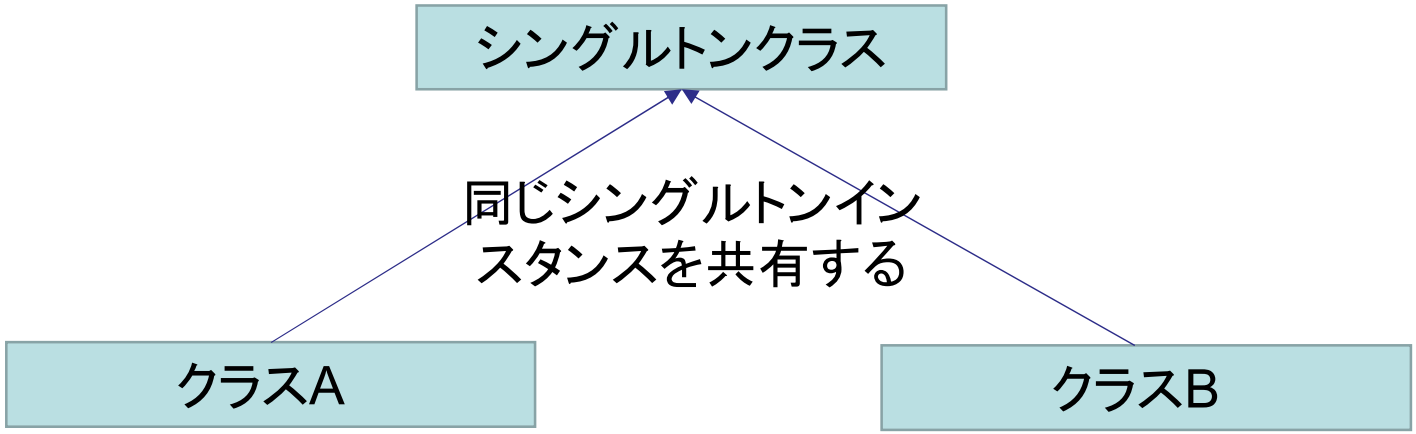
本ステップでおこなうこと

DIコンテナからのインスタンス取得を、index.php以外のクラスから
もできるようにします。



シングルトンクラスの概要(1)

シングルトンクラスとは、アプリケーション全体で、生成されるインスタンスが1つだけのクラスのことです。
アプリケーション全体で、同じインスタンスの状態(＝プロパティ値)を共有することができます。



シングルトンクラスの概要(2)

- 通常のクラスの場合、$aと$bは別インスタンスです。

```
$a = new NormalClass();
$b = new NormalClass();
```

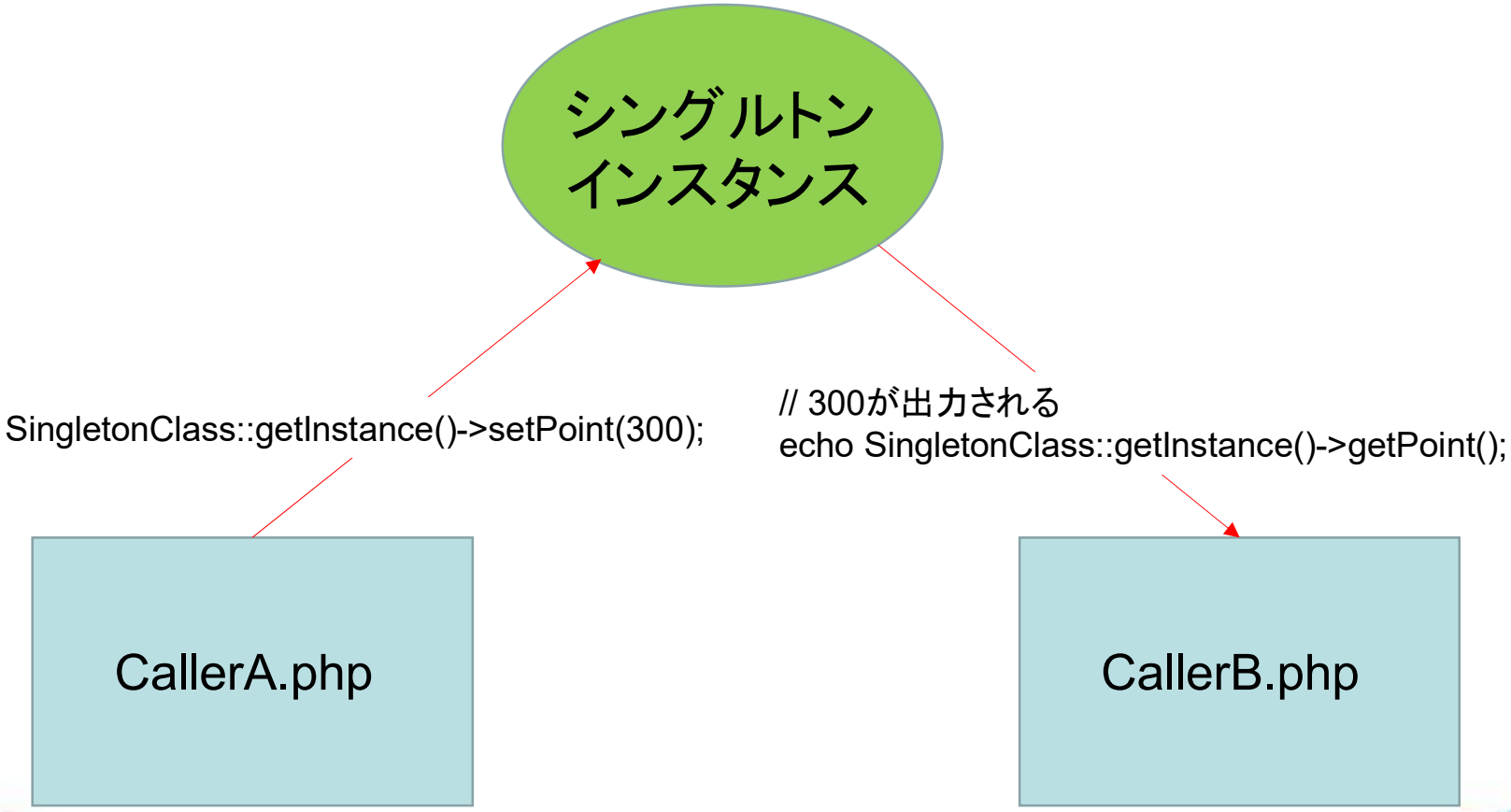
- シングルトンクラスの場合、$aと$bは同じインスタンスです。

```
$a = SingletonClass::getInstance();
$b = SingletonClass::getInstance();
```

※new()ではなくgetInstance()などのstaticメソッド経由でインスタンスを取得します

シングルトンクラスの概要(3)

シングルトンクラスのインスタンスはアプリケーション全体で1つのみですので、そのクラスの状態(プロパティ)も1つのみです。



シングルトンクラスを作ってみる(1)

以下のプログラムで実験してみましょう。

SingletonSample.php

```php
<?php
class SingletonSample
{
    private static $instance;

    public $greeting;

    final public static function getInstance(): object
    {
        if (self::$instance === null) {
            self::$instance = new self();
        }
        return self::$instance;
    }

    final private function __construct()
    {
        ;
    }
}
```

SingletonUserA.php

```php
<?php
require_once __DIR__ . '/SingletonSample.php';
class SingletonUserA
{
    public function setGreeting()
    {
        $singleton = SingletonSample::getInstance();
        $singleton->greeting = 'Hello';
    }
}
```

SingletonUserB.php

```php
<?php
require_once __DIR__ . '/SingletonSample.php';
class SingletonUserB
{
    public function getGreeting()
    {
        $singleton = SingletonSample::getInstance();
        return $singleton->greeting;
    }
}
```

main.php

```php
<?php
require_once __DIR__ . '/SingletonUserA.php';
require_once __DIR__ . '/SingletonUserB.php';

$userA = new SingletonUserA();
$userA->setGreeting();
$userB = new SingletonUserB();
echo $userB->getGreeting(), PHP_EOL;
```

★実行結果
Hello

シングルトンクラスを作ってみる(2)

```php
<?php
class SingletonSample
{
    private static $instance;

    public $greeting;

    final public static function getInstance(): object
    {
        if (self::$instance === null) {
            self::$instance = new self();
        }
        return self::$instance;
    }

    final private function __construct()
    {
        ;
    }
}
```

自身のインスタンスをプロパティとして持つ。インスタンスは1つだけにしたいのでstaticで。

コンストラクタの代わりとなるメソッドを用意する。ここもstaticで。

コンストラクタは用意するが、外部からnewできないよう、privateにする。外部からnewすると、エラーになる。

traitでクラスの断片を挿し込む(1)

トレイトは、クラスの機能の「断片」です。
あるクラスに必要な、一部のプロパティやメソッドを挿し込むことができます。

SomeClass.php

```
require __DIR__ . '/SomeTrait.php';

class SomeClass
{
    use SomeTrait;

    public function sayHello()
    {
        $this->greeting = 'Hello';
        echo $this->getGreeting();
    }
}

$someClass = new SomeClass();
$someClass->sayHello();
```


SomeTrait.php

```
trait SomeTrait
{
    private $greeting;

    private function getGreeting()
    {
        return $this->greeting . ' World !';
    }
}
```

★実行結果
Hello World !

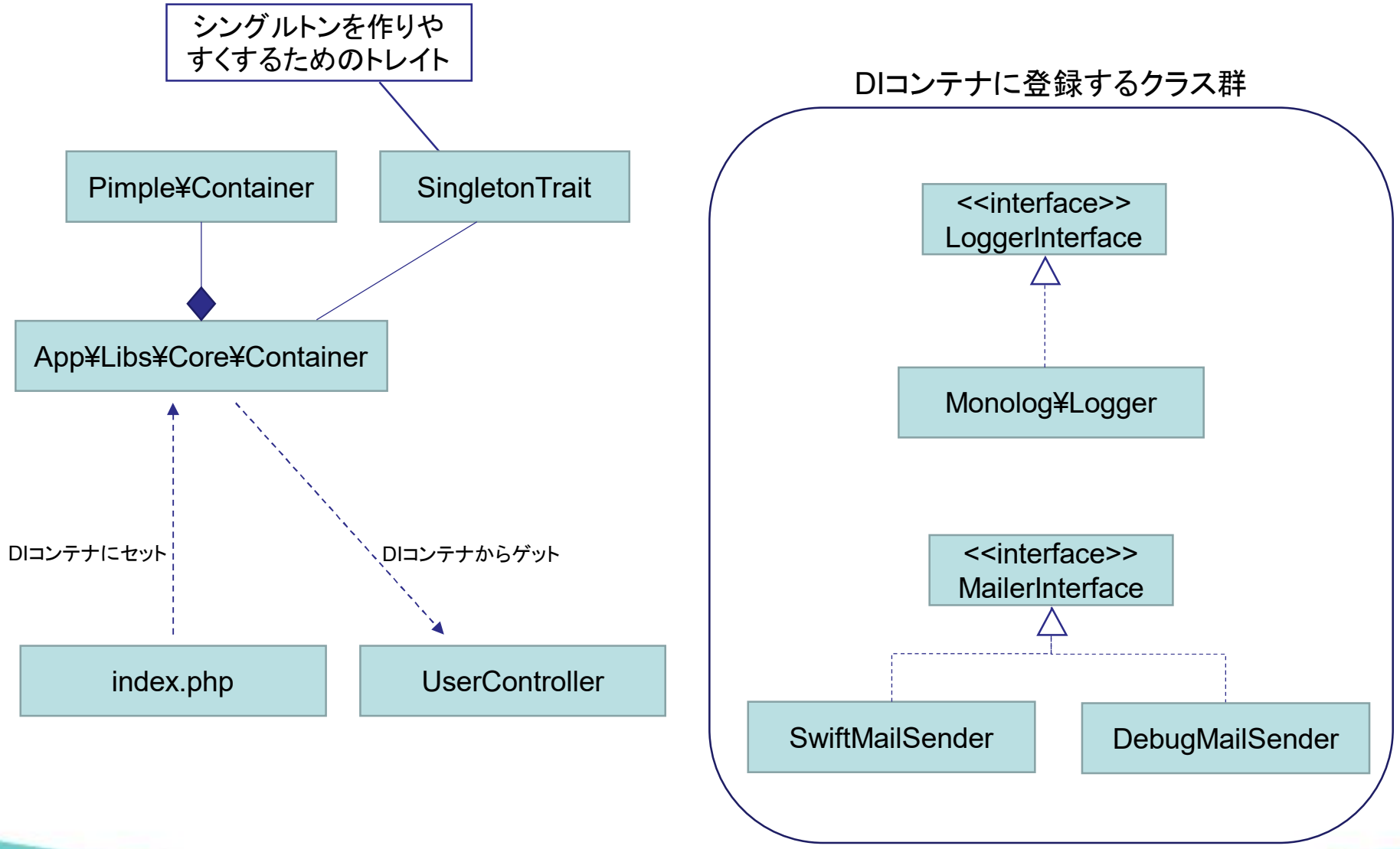
traitでクラスの断片を挿し込む(2)

シングルトンのような決まりきった書き方は、トレイトにしておくと便利です。以下はその例です。
シングルトンに必要なプロパティとメソッドのみを持つトレイトです。

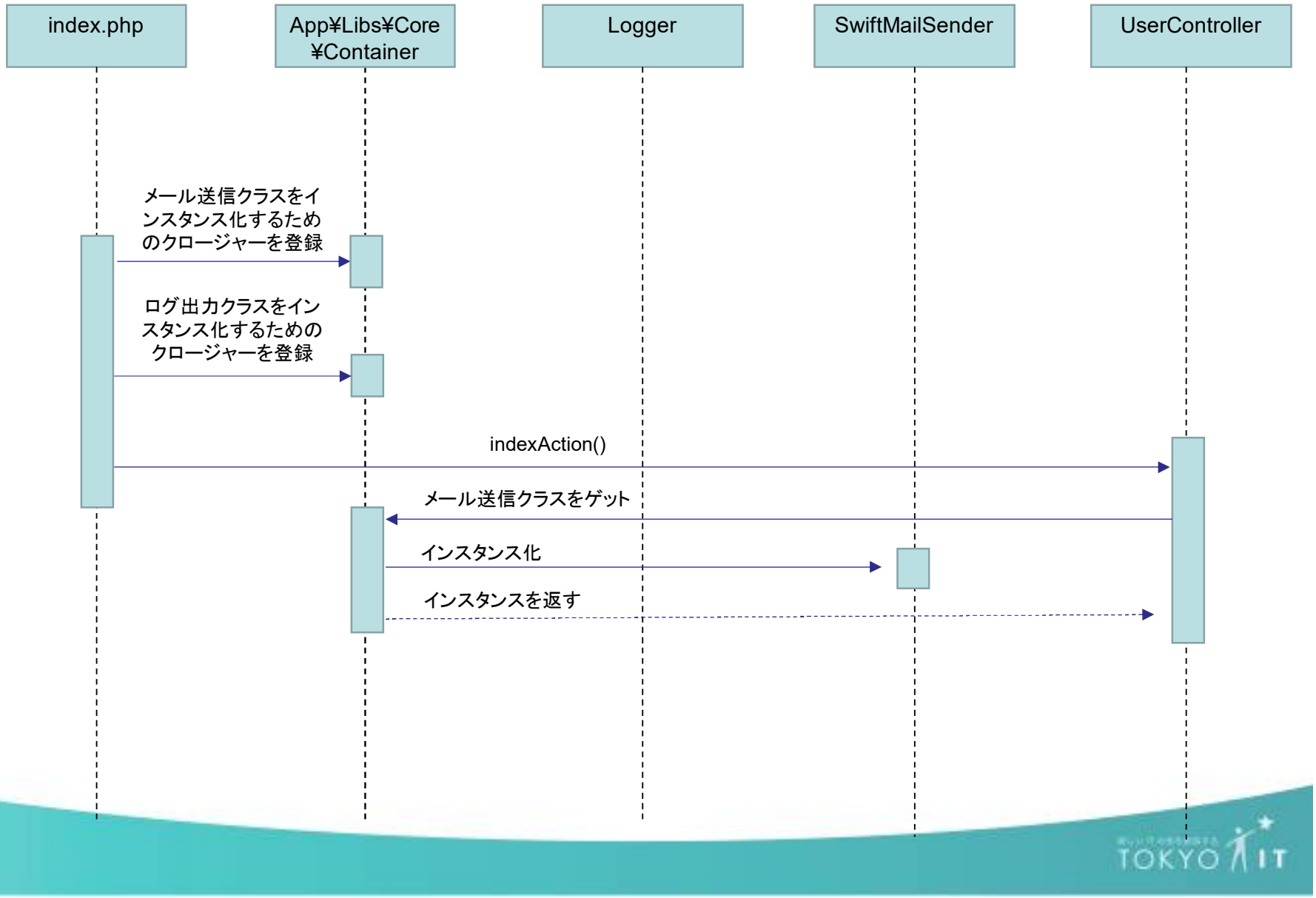
SingletonTrait.php

```
<?php
trait SingletonTrait
{
    private static $instance;

    final public static function getInstance(): object
    {
        if (self::$instance === null) {
            self::$instance = new self();
        }
        return self::$instance;
    }

    final private function __construct()
    {
        ;
    }
}
```

本ステップのクラス構成



本ステップの処理の流れ



本ステップの変更ファイル一覧

●追加したファイル

- app/Libs/Core/Container.php
  → pimpleコンテナのラッパークラス。シングルトンクラス。

- app/Libs/Core/Traits/SingletonTrait.php
  → シングルトンクラスを作りやすくするためのトレイト

●変更したファイル

- public/index.php
  → DIコンテナへの登録処理を変更

- app/Modules/User/Controllers/UserController.php
  → DIコンテナからの取得処理を追加

参考情報

- PHP本格入門(上)
  「3-2-11 インスタンスの状態に左右されないメソッド - 静的メソッド」
- PHP本格入門(上)
  「3-2-12 プログラム処理全体で暮らすの1つの状態を共有する - 静的プロパティ」
- PHP本格入門(上)
  「3-5 横断的で定型的な処理をクラスに挿し込む - トレイト」