

KAI SASSNOWSKI

DEMYSTIFYING DEPENDENCY INJECTION CONTAINERS

DEPENDENCY INJECTION

```
class SessionStorage
{
    public function __construct()
    {
        session_start();
    }

    public function set($key, $value)
    {
        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class SessionStorage
{
    public function __construct()
    {
        session_start();
    }

    public function set($key, $value)
    {
        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class SessionStorage
{
    public function __construct()
    {
        session_start();
    }

    public function set($key, $value)
    {
        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class User
{
    protected $storage;

    function __construct()
    {
        $this->storage = new SessionStorage();
    }

    // ...
}
```

```
class SessionStorage
{
    public function __construct()
    {
        session_start();
    }

    public function set($key, $value)
    {
        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class User
{
    protected $storage;

    function __construct()
    {
        $this->storage = new SessionStorage();
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    // ...
}
```

```
class SessionStorage
{
    public function __construct()
    {
        session_start();
    }

    public function set($key, $value)
    {
        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class User
{
    protected $storage;

    function __construct()
    {
        $this->storage = new SessionStorage();
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}
```

```
class SessionStorage
{
    public function __construct()
    {
        session_start();
    }

    public function set($key, $value)
    {
        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class User
```

```
{
```

```
    protected $storage;
```

```
    function __construct()
```

```
    {
```

```
        $this->storage = new SessionStorage();
```

```
    }
```

```
    function setLanguage($language)
```

```
    {
```

```
        $this->storage->set('language', $language);
```

```
    }
```

```
    function getLanguage()
```

```
    {
```

```
        return $this->storage->get('language');
```

```
    }
```

```
    // ...
```

```
}
```



```
class SessionStorage
{
    public function __construct()
    {
        session_start();
    }

    public function set($key, $value)
    {
        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class User
{
    protected $storage;

    function __construct(SessionStorage $storage)
    {
        $this->storage = $storage;
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}
```

```
class SessionStorage
{
    public function __construct()
    {
        session_start();
    }

    public function set($key, $value)
    {
        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class User
{
    protected $storage;

    function __construct(SessionStorage $storage)
    {
        $this->storage = $storage;
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}
```

```
class SessionStorage
{
    public function __construct()
    {
        session_start();
    }

    public function set($key, $value)
    {
        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class User
{
    protected $storage;

    function __construct(SessionStorage $storage)
    {
        $this->storage = $storage;
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}
```

```
$storage = new SessionStorage();
```

```
$user = new User($storage);
```

```
$user->setLanguage('de');
```

```
$user->getLanguage();
```

```
// => 'de'
```

HOWEVER...

```
$svc = new ShippingService(new ProductLocator(),  
    new PricingService(), new InventoryService(),  
    new TrackingRepository(new ConfigProvider()),  
    new Logger(new EmailLogger(new ConfigProvider())));
```



**CONTAINERS TO
THE RESCUE!**


```
$svc = $container->get(ShippingService::class);
```

MAGIC



3 STAGES OF MAGIC

EUPHORIA

**THIS IS GREAT!
IT JUST WORKS (TM)!
I'M GETTING SO MUCH DONE!**




```
PHP Fatal error: Uncaught
exception
'ReflectionException' with
message 'Class log does not
exist' in /Users/Kai/dev/
laravel/vendor/laravel/
framework/src/Illuminate/
Container/Container.php:776
Stack trace:
#0 /Users/Kai/dev/laravel/
vendor/laravel/framework/
```


DISTRUST

HOW DOES THIS WORK?
IT BROKE, NOW WHAT?
WHAT IS EVEN GOING ON?





~~HOLY BIBLE~~
DI-CONTAINERS

UNDERSTANDING

IT'S NOT MAGIC

IT'S JUST ANOTHER ABSTRACTION

I KNOW WHERE TO LOOK IF THINGS BREAK

**IF YOU UNDERSTAND HOW
THINGS WORK THEY STOP
BEING MAGIC.**

The guy on stage right now

**LET'S WRITE OUR
OWN!**

**A CONTAINER IS LIKE AN
AGGREGATION OF FACTORIES.**

Still the same guy

```
class Container
{
    protected $bindings = [];
}
```

```
class Container
{
    protected $bindings = [];

    public function set($abstract, $concrete)
    {
        $this->bindings[$abstract] = $concrete;
    }
}
```

```
class Container
{
    protected $bindings = [];

    public function set($abstract, callable $concrete)
    {
        $this->bindings[$abstract] = $concrete;
    }
}
```

```
class Container
{
    protected $bindings = [];

    public function set($abstract, callable $concrete)
    {
        $this->bindings[$abstract] = $concrete;
    }
}
```

```
class Container
{
    protected $bindings = [];

    public function set($abstract, callable $concrete)
    {
        $this->bindings[$abstract] = $concrete;
    }

    public function get($abstract)
    {
        return $this->bindings[$abstract]($this);
    }
}
```



```
class Container
{
    protected $bindings = [];

    public function set($abstract, callable $concrete)
    {
        $this->bindings[$abstract] = $concrete;
    }

    public function get($abstract)
    {
        return $this->bindings[$abstract]($this);
    }
}
```

```
class Container
{
    protected $bindings = [];

    public function set($abstract, callable $concrete)
    {
        $this->bindings[$abstract] = $concrete;
    }

    public function get($abstract)
    {
        return $this->bindings[$abstract]($this);
    }
}
```

```
class SessionStorage
{
    public function __construct()
    {
        session_start();
    }

    public function set($key, $value)
    {
        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class SessionStorage
{
    public function __construct()
    {
        session_start();
    }

    public function set($key, $value)
    {
        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class SessionStorage
{
    public function __construct()
    {
        session_start();
    }

    public function set($key, $value)
    {
        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class User
{
    protected $storage;

    function __construct(SessionStorage $storage)
    {
        $this->storage = $storage;
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}
```

```
class SessionStorage
{
    public function __construct()
    {
        session_start();
    }

    public function set($key, $value)
    {
        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class User
{
    protected $storage;

    function __construct(SessionStorage $storage)
    {
        $this->storage = $storage;
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}
```

```
class SessionStorage
```

```
{
```

```
    public function __construct()
```

```
    {
```

```
        session_start();
```

```
    }
```

```
    public function set($key, $value)
```

```
    {
```

```
        $_SESSION[$key] = $value;
```

```
    }
```

```
    public function get($key)
```

```
    {
```

```
        return $_SESSION[$key];
```

```
    }
```

```
}
```

```
class User
{
    protected $storage;

    function __construct(SessionStorage $storage)
    {
        $this->storage = $storage;
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}
```

```

class SessionStorage
{
    protected $logger;

    public function __construct(Logger $logger)
    {
        session_start();

        $this->logger = $logger;
    }

    public function set($key, $value)
    {
        $this->logger->log(
            "Setting [$key] to [$value]"
        );

        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}

```

```

class User
{
    protected $storage;

    function __construct(SessionStorage $storage)
    {
        $this->storage = $storage;
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}

```



```

class SessionStorage
{
    protected $logger;

    public function __construct(Logger $logger)
    {
        session_start();

        $this->logger = $logger;
    }

    public function set($key, $value)
    {
        $this->logger->log(
            "Setting [$key] to [$value]"
        );

        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}

```

```

class User
{
    protected $storage;

    function __construct(SessionStorage $storage)
    {
        $this->storage = $storage;
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}

```

```

class SessionStorage
{
    protected $logger;

    public function __construct(Logger $logger)
    {
        session_start();

        $this->logger = $logger;
    }

    public function set($key, $value)
    {
        $this->logger->log(
            "Setting [$key] to [$value]"
        );

        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}

```

```

class User
{
    protected $storage;

    function __construct(SessionStorage $storage)
    {
        $this->storage = $storage;
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}

```

```

class SessionStorage
{
    protected $logger;

    public function __construct(Logger $logger)
    {
        session_start();

        $this->logger = $logger;
    }

    public function set($key, $value)
    {
        $this->logger->log(
            "Setting [$key] to [$value]"
        );

        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}

```

```

class User
{
    protected $storage;

    function __construct(SessionStorage $storage)
    {
        $this->storage = $storage;
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}

```

```

class SessionStorage
{
    protected $logger;

    public function __construct(Logger $logger)
    {
        session_start();

        $this->logger = $logger;
    }

    public function set($key, $value)
    {
        $this->logger->log(
            "Setting [$key] to [$value]"
        );

        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}

```

```

class User
{
    protected $storage;

    function __construct(SessionStorage $storage)
    {
        $this->storage = $storage;
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}

```

```
class SessionStorage
{
    protected $logger;

    public function __construct(Logger $logger)
    {
        session_start();

        $this->logger = $logger;
    }

    public function set($key, $value)
    {
        $this->logger->log(
            "Setting [$key] to [$value]"
        );

        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class User
{
    protected $storage;

    function __construct(SessionStorage $storage)
    {
        $this->storage = $storage;
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}
```

```

class SessionStorage
{
    protected $logger;

    public function __construct(Logger $logger)
    {
        session_start();

        $this->logger = $logger;
    }

    public function set($key, $value)
    {
        $this->logger->log(
            "Setting [$key] to [$value]"
        );

        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}

```

```

class User
{
    protected $storage;

    function __construct(SessionStorage $storage)
    {
        $this->storage = $storage;
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}

```

class DatabaseLogger implements Logger

```

{
    public function log($message)
    {
        echo "Logging $message to the database..";
    }
}

```

```
class SessionStorage
{
    protected $logger;

    public function __construct(Logger $logger)
    {
        session_start();

        $this->logger = $logger;
    }

    public function set($key, $value)
    {
        $this->logger->log(
            "Setting [$key] to [$value]"
        );

        $_SESSION[$key] = $value;
    }

    public function get($key)
    {
        return $_SESSION[$key];
    }
}
```

```
class User
{
    protected $storage;

    function __construct(SessionStorage $storage)
    {
        $this->storage = $storage;
    }

    function setLanguage($language)
    {
        $this->storage->set('language', $language);
    }

    function getLanguage()
    {
        return $this->storage->get('language');
    }

    // ...
}
```

```
class DatabaseLogger implements Logger
{
    public function log($message)
    {
        echo "Logging $message to the database...";
    }
}
```

```
$container = new Container();
```



```
$container = new Container();
```

```
$container->set(User::class, function () {  
});
```

```
$container = new Container();  
  
$container->set(User::class, function () {  
    return new User();  
});
```

```
$container = new Container();  
  
$container->set(User::class, function () {  
    return new User(???)  
});
```

```
$container = new Container();

$container->set(User::class, function () {
    return new User(
        new SessionStorage(
            new DatabaseLogger()));
});
```

```
$container = new Container();  
  
$container->set(User::class, function () {  
    return new User(???)  
});
```

```
class Container
{
    protected $bindings = [];

    public function set($abstract, callable $concrete)
    {
        $this->bindings[$abstract] = $concrete;
    }

    public function get($abstract)
    {
        return $this->bindings[$abstract]($this);
    }
}
```

```
class Container
{
    protected $bindings = [];

    public function set($abstract, callable $concrete)
    {
        $this->bindings[$abstract] = $concrete;
    }

    public function get($abstract)
    {
        return $this->bindings[$abstract]($this);
    }
}
```

```
$container = new Container();  
  
$container->set(User::class, function () {  
    return new User(???)  
});
```



```
$container = new Container();

$container->set(User::class, function (Container $c) {
    return new User(??);
});
```

```
$container = new Container();

$container->set(User::class, function (Container $c) {
    return new User($c->get(SessionStorage::class));
});
```

```
$container = new Container();
```

```
$container->set(User::class, function (Container $c) {  
    return new User($c->get(SessionStorage::class));  
});
```

```
$container->set(SessionStorage::class, function (Container $c) {  
    return new SessionStorage($c->get(Logger::class));  
});
```

```
$container = new Container();

$container->set(User::class, function (Container $c) {
    return new User($c->get(SessionStorage::class));
});

$container->set(SessionStorage::class, function (Container $c) {
    return new SessionStorage($c->get(Logger::class));
});
```

```
$container = new Container();

$container->set(User::class, function (Container $c) {
    return new User($c->get(SessionStorage::class));
});

$container->set(SessionStorage::class, function (Container $c) {
    return new SessionStorage($c->get(Logger::class));
});

$container->set(Logger::class, function (Container $c) {
    return new DatabaseLogger();
});
```

```
$container = new Container();

$container->set(User::class, function (Container $c) {
    return new User($c->get(SessionStorage::class));
});

$container->set(SessionStorage::class, function (Container $c) {
    return new SessionStorage($c->get(Logger::class));
});

$container->set(Logger::class, function (Container $c) {
    return new DatabaseLogger();
});

$user = $container->get(User::class);
$user->setLanguage('de');
$user->getLanguage();
// => 'de'
```

```
$container->get(User::class);
```

```
$container->get(User::class);  
  
function (Container $c) {  
    return new User($c->get(SessionStorage::class));  
};
```



```
$container->get(User::class);
```

```
function (Container $c) {  
    return new User($c->get(SessionStorage::class));  
};
```

```
$container->get(SessionStorage::class)
```

```
$container->get(User::class);
```

```
function (Container $c) {  
    return new User($c->get(SessionStorage::class));  
};
```

```
$container->get(SessionStorage::class)
```

```
function (Container $c) {  
    return new SessionStorage($c->get(Logger::class));  
}
```

```
$container->get(User::class);
```

```
function (Container $c) {  
    return new User($c->get(SessionStorage::class));  
};
```

```
$container->get(SessionStorage::class)
```

```
function (Container $c) {  
    return new SessionStorage($c->get(Logger::class));  
}
```

```
$container->get(Logger::class)
```

```
$container->get(User::class);  
  
function (Container $c) {  
    return new User($c->get(SessionStorage::class));  
};
```

```
$container->get(SessionStorage::class)  
  
function (Container $c) {  
    return new SessionStorage($c->get(Logger::class));  
}
```

```
$container->get(Logger::class)  
  
function (Container $c) {  
    return new DatabaseLogger();  
}
```

```
$container->get(User::class);
```

```
function (Container $c) {  
    return new User($c->get(SessionStorage::class));  
};
```

```
$container->get(SessionStorage::class)
```

```
function (Container $c) {  
    return new SessionStorage($c->get(Logger::class));  
}
```

```
DatabaseLogger();
```

```
$container->get(User::class);
```

```
function (Container $c) {  
    return new User($c->get(SessionStorage::class));  
};
```

```
$container->get(SessionStorage::class)
```

```
function (Container $c) {  
    return new SessionStorage(DatabaseLogger());  
}
```

```
$container->get(User::class);  
  
function (Container $c) {  
    return new User($c->get(SessionStorage::class));  
};
```

```
SessionStorage(DatabaseLogger())
```

```
$container->get(User::class);  
  
function (Container $c) {  
    return new User(SessionStorage(DatabaseLogger()));  
};
```



```
User(SessionStorage(DatabaseLogger()));
```

```
User(SessionStorage(DatabaseLogger()));
```

```
class Container
{
    protected $bindings = [];

    public function set($abstract, callable $concrete)
    {
        $this->bindings[$abstract] = $concrete;
    }

    public function get($abstract)
    {
        return $this->bindings[$abstract]($this);
    }
}
```

AUTOWIRING

**THE ABILITY OF THE CONTAINER TO
AUTOMATICALLY BUILD A CLASS
WITHOUT NEEDING AN EXPLICIT
BINDING.**

Take a guess

```
$container = new Container();

$container->set(User::class, function (Container $c) {
    return new User($c->get(SessionStorage::class));
});

$container->set(SessionStorage::class, function (Container $c) {
    return new SessionStorage($c->get(Logger::class));
});

$container->set(Logger::class, function (Container $c) {
    return new DatabaseLogger();
});

$user = $container->get(User::class);
$user->setLanguage('de');
$user->getLanguage();
// => 'de'
```

```
$container = new Container();

$container->set(User::class, function (Container $c) {
    return new User($c->get(SessionStorage::class));
});

$container->set(SessionStorage::class, function (Container $c) {
    return new SessionStorage($c->get(Logger::class));
});

$container->set(Logger::class, function (Container $c) {
    return new DatabaseLogger();
});

$user = $container->get(User::class);
$user->setLanguage('de');
$user->getLanguage();
// => 'de'
```

```
$container = new Container();

$container->set(User::class, function (Container $c) {
    return new User($c->get(SessionStorage::class));
});

$container->set(SessionStorage::class, function (Container $c) {
    return new SessionStorage($c->get(DatabaseLogger::class));
});

$container->set(Logger::class, function (Container $c) {
    return new DatabaseLogger();
});

$user = $container->get(User::class);
$user->setLanguage('de');
$user->getLanguage();
// => 'de'
```



```
$container = new Container();

$container->set(User::class, function (Container $c) {
    return new User($c->get(SessionStorage::class));
});

$container->set(SessionStorage::class, function (Container $c) {
    return new SessionStorage($c->get(DatabaseLogger::class));
});

$container->set(DatabaseLogger::class, function (Container $c) {
    return new DatabaseLogger();
});

$user = $container->get(User::class);
$user->setLanguage('de');
$user->getLanguage();
// => 'de'
```

```
$container = new Container();  
  
$user = $container->get(User::class);  
$user->setLanguage('de');  
$user->getLanguage();  
// => 'de'
```

REFLECTION BASED AUTOWIRING

- ▶ Build a ReflectionClass
- ▶ Build its dependencies
 - ▶ Get the constructor
 - ▶ Get its parameters
 - ▶ Look at the types
 - ▶ Recursively build the dependencies
- ▶ Create new instance

1. BUILD A REFLECTION CLASS

```
public function get($abstract)
{
    $reflection = new ReflectionClass($abstract);
}
```

2. BUILD ITS DEPENDENCIES

```
public function get($abstract)
{
    $reflection = new ReflectionClass($abstract);
}
```

2. BUILD ITS DEPENDENCIES

```
public function get($abstract)
{
    $reflection = new ReflectionClass($abstract);
    $dependencies = $this->buildDependencies($reflection);
}
```

2.1. GET ITS CONSTRUCTOR

```
public function get($abstract)
{
    $reflection = new ReflectionClass($abstract);
    $dependencies = $this->buildDependencies($reflection);
}
```

2.1. GET ITS CONSTRUCTOR

```
private function buildDependencies($reflection)
{
    $constructor = $reflection->getConstructor();
}
```


2.1. GET ITS CONSTRUCTOR

```
private function buildDependencies($reflection)
{
    if (!$constructor = $reflection->getConstructor()) {
        return [];
    }
}
```

2.2. GET ITS PARAMETERS

```
private function buildDependencies($reflection)
{
    if (!$constructor = $reflection->getConstructor()) {
        return [];
    }
}
```

2.2. GET ITS PARAMETERS

```
private function buildDependencies($reflection)
{
    if (!$constructor = $reflection->getConstructor()) {
        return [];
    }

    $params = $constructor->getParameters();
}
```

2.2. GET ITS PARAMETERS

```
private function buildDependencies($reflection)
{
    if (!$constructor = $reflection->getConstructor()) {
        return [];
    }

    $params = $constructor->getParameters();

    return array_map(function ($param) {

    }, $params);
}
```

2.3. LOOK AT THE TYPES

```
private function buildDependencies($reflection)
{
    if (!$constructor = $reflection->getConstructor()) {
        return [];
    }

    $params = $constructor->getParameters();

    return array_map(function ($param) {

    }, $params);
}
```

2.3. LOOK AT THE TYPES

```
private function buildDependencies($reflection)
{
    if (!$constructor = $reflection->getConstructor()) {
        return [];
    }

    $params = $constructor->getParameters();

    return array_map(function ($param) {
        $type = $param->getType();

    }, $params);
}
```

2.3. LOOK AT THE TYPES

```
private function buildDependencies($reflection)
{
    if (!$constructor = $reflection->getConstructor()) {
        return [];
    }

    $params = $constructor->getParameters();

    return array_map(function ($param) {
        if (!$type = $param->getType()) {
            throw new RuntimeException();
        }
    }, $params);
}
```

2.4. RECURSIVELY BUILD THE DEPENDENCY

```
private function buildDependencies($reflection)
{
    if (!$constructor = $reflection->getConstructor()) {
        return [];
    }

    $params = $constructor->getParameters();

    return array_map(function ($param) {
        if (!$type = $param->getType()) {
            throw new RuntimeException();
        }
    }, $params);
}
```


2.4. RECURSIVELY BUILD THE DEPENDENCY

```
private function buildDependencies($reflection)
{
    if (!$constructor = $reflection->getConstructor()) {
        return [];
    }

    $params = $constructor->getParameters();

    return array_map(function ($param) {
        if (!$type = $param->getType()) {
            throw new RuntimeException();
        }

        return $this->get($type);
    }, $params);
}
```

```
public function get($abstract)
{
    if (isset($this->bindings[$abstract])) {
        return $this->bindings[$abstract]($this);
    }

    $reflection = new ReflectionClass($abstract);

    $dependencies = $this->buildDependencies($reflection);

    return $reflection->newInstanceArgs($dependencies);
}
```

```
$container = new Container();  
  
$user = $container->get(User::class);  
$user->setLanguage('de');  
$user->getLanguage();  
// => 'de'
```

TL;DL RECURSION

THANKS!