Note to self: Delete this slide

KAI SASSNOWSKI / @WARSH33P

DEMYSTIFYING DEPENDENCY INJECTION CONTAINERS

106

THANKS!

KAI SASSNOWSKI / @WARSH33P

DEMYSTIFYING DEPENDENCY INJECTION CONTAINERS

DEPENDENCY INJECTION

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

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

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

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

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

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

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

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

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

```php
class User
{
    protected $storage;

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

    // ...
}
```

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

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

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

```php
class User
{
    protected $storage;

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

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

    // ...
}
```

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

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

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

```php
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');
    }

    // ...
}
```

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

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

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

```php
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');
    }

    // ...
}
```

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

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

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

```php
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');
    }

    // ...
}
```

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

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

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

```php
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');
    }

    // ...
}
```

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

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

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

```php
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');
    }

    // ...
}
```

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

$user = new User($storage);

$user->setLanguage(‘de');

$user->getLanguage();
// => 'de'
```

HOWEVER...

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

CONTAINERS TO THE RESCUE!

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

LET'S WRITE OUR OWN!

A CONTAINER IS LIKE AN AGGREGATION OF FACTORIES.

The guy on stage right now

```php
class Container
{
  private $bindings = [];
}
```

```php
class Container
{
  private $bindings = [];

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

```php
class Container
{
  private $bindings = [];

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

```php
class Container
{
  private $bindings = [];

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

```php
class Container
{
  private $bindings = [];

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

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

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

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

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

```php
class Container
{
  private $bindings = [];

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

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

```php
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];
    }
}
```

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

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

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

```php
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');
    }

    // ...
}
```

```php
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');
    }

    // ...
}
```

```php
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');
    }

    // ...
}
```

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

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

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

```php
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');
    }

    // ...
}
```

```php
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;
    }

   // ...
}
```

```php
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');
    }

    // ...
}
```

```php
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;
    }

   // ...
}
```

```php
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');
    }

    // ...
}
```

```php
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');
    }

    // ...
}
```

```php
class Logger
{
    public function log($message)
    {
        echo "Logging $message to the database…";
    }
}
```

```php
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…";
    }
}
```

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

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

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

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

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

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

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

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

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

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

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

```php
class Container
{
   private $bindings = [];

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

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

```php
class Container
{
  private $bindings = [];

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

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

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

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

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

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

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

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

```php
$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));
});
```

```php
$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 Logger();
});
```

```php
$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 Logger();
});

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```php
Logger();
```

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

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

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

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

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

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

```
SessionStorage(Logger());
```

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

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

```
User(SessionStorage(Logger()));
User(SessionStorage(Logger()));
```

```php
class Container
{
   private $bindings = [];

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

   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.

Same guy

```php
$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 Logger();
});

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

```php
$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

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

2. BUILD ITS DEPENDENCIES

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

2. BUILD ITS DEPENDENCIES

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

2.1. GET ITS CONSTRUCTOR

```php
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

```php
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);

}
```

```php
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);
}
```

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

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

TL;DL RECURSION

THANKS!