Real World Dependency Injection

Stephan Hochdörfer, bitExpert AG
Real World Dependency Injection

About me

- Stephan Hochdörfer
- Head of IT at bitExpert AG, Germany
- enjoying PHP since 1999
- S.Hochdoerfer@bitExpert.de
- @shochdoerfer
Real World Dependency Injection

I so <3 Dependency Injection.
Real World Dependency Injection

Separation of Concerns
Design by Contract

Real World Dependency Injection
Real World Dependency Injection

Dependency Injection
Dependency Injection

"[...] pattern that allows the removal of hard-coded dependencies and makes it possible to change them, whether at run-time or compile-time." (Wikipedia)
Real World Dependency Injection

What are Dependencies?
Real World Dependency Injection

Are dependencies bad?
Real World Dependency Injection

Dependencies: Tight coupling
Real World Dependency Injection

No code reuse!
No isolation, not testable!
Real World Dependency Injection

Dependency hell!
What is Dependency Injection?
What is Dependency Injection?

```java
new DeletePage(new PageManager());
```
Real World Dependency Injection

What is Dependency Injection?

Consumer
What is Dependency Injection?
What is Dependency Injection?
What is Dependency Injection?

Consumer

Dependencies

Container
Real World Dependency Injection

„new“ is evil!
Real World Dependency Injection

„new“ is evil!

```php
<?php

class DeletePage extends Mvc_Action_AAction {

    private $pageManager;

    public function __construct() {
        $this->pageManager = new PageManager();
    }

    protected function execute(Mvc_Request $request) {
        $this->pageManager->delete(
            (int) $request->get('pageId')
        );
    }
}
```
„new“ is evil!

```php
<?php

class DeletePage extends Mvc_Action_AAction {
    private $pageManager;

    public function __construct(PageManager $pm) {
        $this->pageManager = $pm;
    }

    protected function execute(Mvc_Request $request) {
        $this->pageManager->delete(
            (int) $request->get('pageId')
        );
    }
}
```
"High-level modules should not depend on low-level modules. Both should depend on abstractions."

Robert C. Martin
„new“ is evil!

```php
<?php

class DeletePage extends Mvc_Action_AAction {

    private $pageManager;

    public function __construct(IPageManager $pm) {
        $this->pageManager = $pm;
    }

    protected function execute(Mvc_Request $request) {
        $this->pageManager->delete((int) $request->get('pageId'));
    }

}
```
Real World Dependency Injection

How to manage Dependencies?
How to manage Dependencies?

Simple container vs. Full stacked DI Framework
The container acts as a glue point!
Real World Dependency Injection

How to inject dependencies?
Real World Dependency Injection

Constructor Injection

```php
<?php

class MySampleService implements IMySampleService {
    /**
     * @var ISampleDao
     */
    private $sampleDao;

    public function __construct(ISampleDao $sampleDao) {
        $this-&gt;sampleDao = $sampleDao;
    }
}
```
Real World Dependency Injection

Setter Injection

```php
<?php

class MySampleService implements IMySampleService {
    /**
     * @var ISampleDao
     */
    private $sampleDao;

    public function setSampleDao(ISampleDao $sampleDao) {
        $this->sampleDao = $sampleDao;
    }
}
```
Interface Injection

<?php

interface IApplicationContextAware {
    public function setCtx(IApplicationContext $ctx);
}

Real World Dependency Injection
Interface Injection

```php
<?php

class MySampleService implements IMySampleService, IApplicationContextAware {
    /**
     * @var IApplicationContext
     */
    private $ctx;

    public function setCtx(IApplicationContext $ctx) {
        $this->ctx = $ctx;
    }
}
```
Real World Dependency Injection

Property Injection
Real World Dependency Injection

Property Injection

"NEIN NEIN NEIN NEIN!"

David Zülke
Real World Dependency Injection

How to wire it all up?
Real World Dependency Injection

Annotations

```php
<?php

class MySampleService implements IMySampleService {
    private $sampleDao;

    /**
     * @Inject
     */
    public function __construct(ISampleDao $sampleDao)
    {
        $this->sampleDao = $sampleDao;
    }
}
```
Real World Dependency Injection

Annotations

```php
<?php

class MySampleService implements IMySampleService {
    private $sampleDao;

    /**
     * @Inject
     * @Named('TheSampleDao')
     */
    public function __construct(ISampleDao $sampleDao)
    {
        $this->sampleDao = $sampleDao;
    }
}
```
<?xml version="1.0" encoding="UTF-8" ?>
<beans>
    <bean id="SampleDao" class="SampleDao">
        <constructor-arg value="app_sample" />
        <constructor-arg value="iSampleId" />
        <constructor-arg value="BoSample" />
    </bean>

    <bean id="SampleService" class="MySampleService">
        <constructor-arg ref="SampleDao" />
    </bean>
</beans>
services:
  SampleDao:
    class: SampleDao
    arguments: ['app_sample', 'iSampleId', 'BoSample']
  SampleService:
    class: SampleService
    arguments: [@SampleDao]
<?php

class BeanCache extends Beanfactory_Container_PHP {
    protected function createSampleDao() {
        $oBean = new SampleDao('app_sample', 'iSampleId', 'BoSample');
        return $oBean;
    }

    protected function createMySampleService() {
        $oBean = new MySampleService($this->getBean('SampleDao'));
        return $oBean;
    }
}

Real World Dependency Injection
Real World Dependency Injection

Internal vs. external configuration
Internal vs. external configuration

Class configuration vs. Instance configuration
Why use DI in my daily work?
Unittesting made easy
<?php
require_once 'PHPUnit/Framework.php';

class ServiceTest extends PHPUnit_Framework_TestCase {
    public function testSampleService() {
        // set up dependencies
        $sampleDao = $this->getMock('ISampleDao');
        $service = new MySampleService($sampleDao);

        // run test case
        $return = $service->doWork();

        // check assertions
        $this->assertTrue($return);
    }
}

Real World Dependency Injection
One class, multiple configurations
One class, multiple configurations

Real World Dependency Injection

- Page Exporter
  - Released / Published Pages
  - Workingcopy Pages
One class, multiple configurations

```php
<?php
abstract class PageExporter {
    protected function setPageDao(IPageDao $pageDao) {
        $this->pageDao = $pageDao;
    }
}
```
Real World Dependency Injection

One class, multiple configurations

```php
abstract class PageExporter {
    protected function setPageDao(IPageDao $pageDao) {
        $this->pageDao = $pageDao;
    }
}
```

Remember: The contract!
Real World Dependency Injection

One class, multiple configurations

```php
<?php

class PublishedPageExporter extends PageExporter {
    public function __construct() {
        $this->setPageDao(new PublishedPageDao());
    }
}

class WorkingCopyPageExporter extends PageExporter {
    public function __construct() {
        $this->setPageDao(new WorkingCopyPageDao());
    }
}
```
One class, multiple configurations

"Only deleted code is good code!"

Oliver Gierke
One class, multiple configurations

```php
<?php

class PageExporter {
    public function __construct(IPageDao $pageDao) {
        $this->pageDao = $pageDao;
    }
}
```
One class, multiple configurations

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<beans>
    <bean id="ExportLive" class="PageExporter">
        <constructor-arg ref="PublishedPageDao" />
    </bean>

    <bean id="ExportWorking" class="PageExporter">
        <constructor-arg ref="WorkingCopyPageDao" />
    </bean>
</beans>
```
Real World Dependency Injection

One class, multiple configurations

```php
// create ApplicationContext instance
$ctx = new ApplicationContext();

// retrieve live exporter
$exporter = $ctx->getBean('ExportLive');

// retrieve working copy exporter
$exporter = $ctx->getBean('ExportWorking');
```
Mocking external service access
Mocking external service access

Booking service -> WS-Connector -> Webservice
Real World Dependency Injection

Mocking external service access

Booking service \(\rightarrow\) WS-Connector \(\rightarrow\) Webservice

Remember: The contract!
Real World Dependency Injection

Mocking external service access

Booking service → FS-Connector → Filesystem
Mocking external service access

Booking service → FS-Connector → Filesystem

fullfills the contact!
Real World Dependency Injection

Clean, readable code
<?php

class DeletePage extends Mvc_Action_AAction {
    private $pageManager;

    public function __construct(IPageManager $pm) {
        $this->pageManager = $pm;
    }

    protected function execute(Mvc_Request $request) {
        $this->pageManager->delete((int) $request->get('pageId'));

        return new ModelAndView($this->getSuccessView());
    }
}
Real World Dependency Injection

No framework dependencies
<?php

class MySampleService implements IMySampleService {
    private $sampleDao;

    public function __construct(ISampleDao $sampleDao) {
        $this->sampleDao = $sampleDao;
    }

    public function getSample($sampleId) {
        try {
            return $this->sampleDao->readById($sampleId);
        } catch(DaoException $exception) {}
    }
}
Real World Dependency Injection

Getting rid of the "noise"
Real World Dependency Injection

Increase code reuse!
Real World Dependency Injection

Helps to understand the code!
Real World Dependency Injection

Brings back the fun again :)
Real World Dependency Injection

No standard. No tooling support.
It takes some time to understand DI.
Real World Dependency Injection

Configuration vs. Runtime
Real World Dependency Injection

DI is not slow!
Thank you!