Modular Application Architecture

Confoo.ca 2012

Tobias Schlitt (@tobySen)

March 2nd 2012

About me

- Degree in computer sience

About me

- Degree in computer sience
- More than 10 years of professional PHP

About me

- Degree in computer sience
- More than 10 years of professional PHP
- Open source enthusiasts
- Contributing to various FLOSS projects

Co-founder of

Co-founder of

Helping people to create high quality web applications.

Co-founder of

Qafoo
passion for software quality

Helping people to create high quality web applications.

- Expert consulting
- Individual training
- …

Co-founder of

passion for software quality

Helping people to create high quality web applications.

- Expert consulting
- Individual training
- . . .

- Software architecture
- OO design
- Quality assurance
- . . .

Co-founder of

passion for software quality

Helping people to create high quality web applications.

- Expert consulting
- Individual training
- . . .

- Software architecture
- OO design
- Quality assurance
- . . .

`http://qafoo.com`

Outline

Motivation

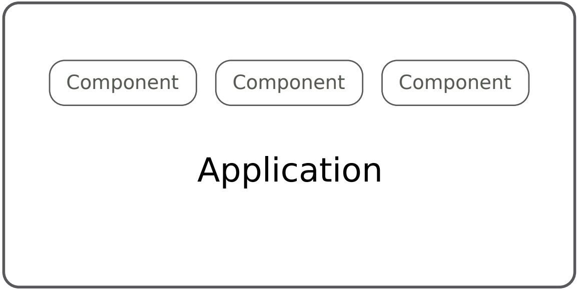
Resources

Approaches

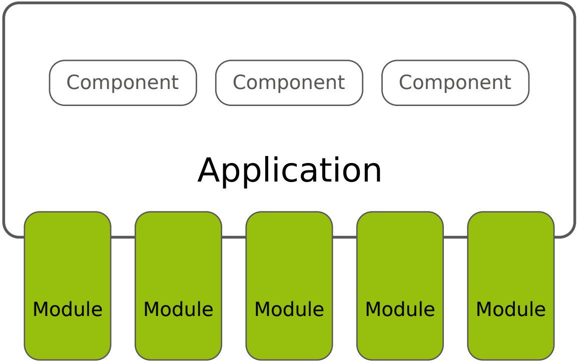
Real world

Summary

Application



Modules


Component
Component
Component
Application
Module
Module
Module
Module
Module

Why modules?

- Need for customization
  - Custom setup for customers
  - 3rd party extensions

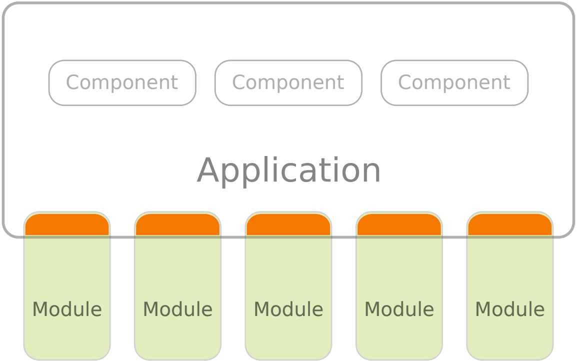
Why modules?

- Need for customization
  - Custom setup for customers
  - 3rd party extensions
- Develop modules seperately from main application
  - External developers
  - Seperate release cycles

Why modules?

- Need for customization
  - Custom setup for customers
  - 3rd party extensions
- Develop modules seperately from main application
  - External developers
  - Seperate release cycles
- Slag the main application
  - Raise maintainability

Essential


Component
Component
Component
Application
Module
Module
Module
Module
Module

Challenges

- Module structure
- Registration / configuration
- Handling ressources
- Interaction with core

Challenges

- Module structure ✓
- Registration / configuration
- Handling ressources
- Interaction with core

Challenges

- Module structure ✓
- Registration / configuration ✓
- Handling ressources
- Interaction with core

Challenges

- Module structure ✓
- Registration / configuration ✓
- Handling ressources ⟶
- Interaction with core

Challenges

- Module structure ✓
- Registration / configuration ✓
- Handling ressources ⟶
- Interaction with core ⟹

Outline

Motivation

Resources

Approaches

Real world

Summary

Dealing with resources

- Typical module resources
  - Templates
  - Translations
  - Images
  - CSS

Dealing with resources

- Typical module resources
  - Templates
  - Translations
  - Images
  - CSS
- Resources handled by code are “easy”
  - Register “overrides“

Dealing with resources

- Typical module resources
  - Templates
  - Translations
  - Images
  - CSS
- Resources handled by code are “easy”
  - Register “overrides“
- Static file resources are not
  - Put modules in a web accessible path?
  - Copy / link static files to `htdocs/`?
  - Pipe static files through PHP?
  - Webserver configuration?

Outline

Motivation

Resources

Approaches

Real world

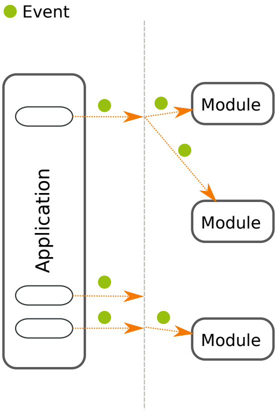
Summary

Outline

- Approaches
  - Event handling
  - Data handling

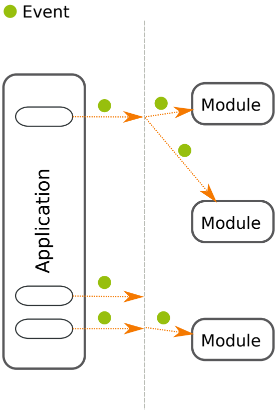
Event handling

- Interaction



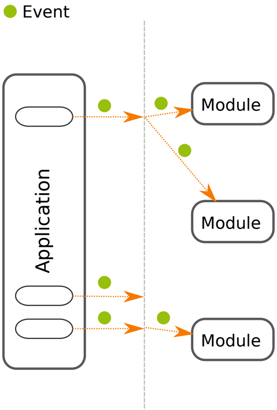
Event handling

- Interaction
- Modules register for event types



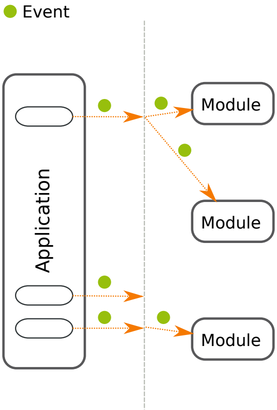
Event handling

- Interaction
- Modules register for event types
- Events “fired” (by core or module)



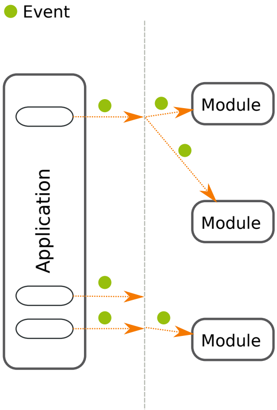
Event handling

- Interaction
- Modules register for event types
- Events “fired” (by core or module)
- Registered modules informed
- Maybe including data



Event handling

- Interaction
- Modules register for event types
- Events “fired” (by core or module)
- Registered modules informed
- Maybe including data
- Optionally transparent



Subject-Observer

```php
<?php

class Subject
{
    public function doSomething()
    {
        $this->notify( 'doSomethingStart' );
        // ...
        $this->notify( 'doSomethingEnd' );
    }
}
```

Subject-Observer

```php
<?php

class Subject
{
    protected $observers = array();

    public function addObserver( Observer $observer )
    {
        $this->observers[] = $observer;
    }

    public function notify( $event, $data = null )
    {
        foreach ( $this->observers as $observer )
        {
            $observer->$event( $data );
        }
    }

    public function doSomething()
    {
        $this->notify( 'doSomethingStart' );
        // ...
        $this->notify( 'doSomethingEnd' );
    }
}
```

Subject-Observer

```php
<?php

class Subject
{
    protected $observers = array();

    public function addObserver( Observer $observer )
    {
        $this->observers[] = $observer;
    }

    public function notify( $event, $data = null )
    {
        foreach ( $this->observers as $observer )
        {
            $observer->$event( $data );
        }
    }

    public function doSomething()
    {
        $this->notify( 'doSomethingStart' );
        // ...
        $this->notify( 'doSomethingEnd' );
    }
}
```

Subject-Observer

```php
<?php

class Observer
{
    public function doSomethingStart()
    {
        // ...
    }

    public function doSomethingEnd()
    {
        // ...
    }
}
```

Pro & Contra

- Benefits:

- Drawbacks:

Pro & Contra

- Benefits:
  - Transparent – any number of observers can register
- Drawbacks:

Pro & Contra

- Benefits:
  - Transparent – any number of observers can register
  - (Documented) clearly defined extension API
- Drawbacks:

Pro & Contra

- Benefits:
  - Transparent – any number of observers can register
  - (Documented) clearly defined extension API
  - . . . optionally with clearly defined transmitted data structs
- Drawbacks:

Pro & Contra

- Benefits:
  - Transparent – any number of observers can register
  - (Documented) clearly defined extension API
  - . . . optionally with clearly defined transmitted data structs
- Drawbacks:
  - Fully transparent – you have no idea how long a singal will take to process

Pro & Contra

- Benefits:
  - Transparent – any number of observers can register
  - (Documented) clearly defined extension API
  - … optionally with clearly defined transmitted data structs
- Drawbacks:
  - Fully transparent – you have no idea how long a singal will take to process
  - Limited to defined extension points

Pro & Contra

- Benefits:
  - Transparent – any number of observers can register
  - (Documented) clearly defined extension API
  - . . . optionally with clearly defined transmitted data structs
- Drawbacks:
  - Fully transparent – you have no idea how long a singal will take to process
  - Limited to defined extension points
  - **Requires implementation in each subject**

Signal slot

```php
<?php

$handler = new SignalSlot();

$handler->register(
    'UserLoggedIn',
    array( new LoginCounterModule(), 'handleUserLoggedIn' )
);
$handler->register(
    'UserLoggedIn',
    array( new OnlineUsersModule(), 'updatedUserList' )
);

// In login process
$handler->emit(
    'UserLoggedIn',
    new UserLoggedInData( /* ... */ )
);
```

Signal slot

```php
<?php

$handler = new SignalSlot();

$handler->register(
    'UserLoggedIn',
    array( new LoginCounterModule(), 'handleUserLoggedIn' )
);
$handler->register(
    'UserLoggedIn',
    array( new OnlineUsersModule(), 'updatedUserList' )
);

// In login process
$handler->emit(
    'UserLoggedIn',
    new UserLoggedInData( /* ... */ )
);
```

Signal slot

```php
<?php

$handler = new SignalSlot();

$handler->register(
    'UserLoggedIn',
    array( new LoginCounterModule(), 'handleUserLoggedIn' )
);
$handler->register(
    'UserLoggedIn',
    array( new OnlineUsersModule(), 'updatedUserList' )
);

// In login process
$handler->emit(
    'UserLoggedIn',
    new UserLoggedInData( /* ... */ )
);
```

Signal slot

```php
<?php

$handler = new SignalSlot();

$handler->register(
    'UserLoggedIn',
    array( new LoginCounterModule(), 'handleUserLoggedIn' )
);
$handler->register(
    'UserLoggedIn',
    array( new OnlineUsersModule(), 'updatedUserList' )
);

// In login process
$handler->emit(
    'UserLoggedIn',
    new UserLoggedInData( /* ... */ )
);
```

Pro & Contra

- Benefits:

- Drawbacks:

Pro & Contra

- Benefits:
  - Fully transparent – nobody needs to know who is called
- Drawbacks:

Pro & Contra

- Benefits:
  - Fully transparent – nobody needs to know who is called
  - (Documented) clearly defined extension API
- Drawbacks:

Pro & Contra

- Benefits:
  - Fully transparent – nobody needs to know who is called
  - (Documented) clearly defined extension API
  - . . . optionally with clearly defined transmitted data structs
- Drawbacks:

Pro & Contra

- Benefits:
  - Fully transparent – nobody needs to know who is called
  - (Documented) clearly defined extension API
  - . . . optionally with clearly defined transmitted data structs
  - Can easily be made asynchronous
- Drawbacks:

Pro & Contra

- Benefits:
  - Fully transparent – nobody needs to know who is called
  - (Documented) clearly defined extension API
  - . . . optionally with clearly defined transmitted data structs
  - Can easily be made asynchronous
- Drawbacks:
  - Fully transparent – you have no idea how long a signal will take to process

Pro & Contra

- Benefits:
  - Fully transparent – nobody needs to know who is called
  - (Documented) clearly defined extension API
  - . . . optionally with clearly defined transmitted data structs
  - Can easily be made asynchronous
- Drawbacks:
  - Fully transparent – you have no idea how long a signal will take to process
  - Limited to defined extension points

Pro & Contra

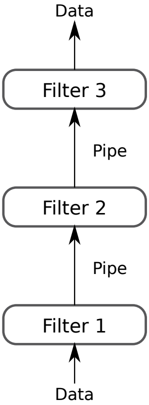
- Benefits:
  - Fully transparent – nobody needs to know who is called
  - (Documented) clearly defined extension API
  - . . . optionally with clearly defined transmitted data structs
  - Can easily be made asynchronous
- Drawbacks:
  - Fully transparent – you have no idea how long a signal will take to process
  - Limited to defined extension points
  - Signal-Slot handler needs to be injected

Outline

- Approaches
  - Event handling
  - Data handling

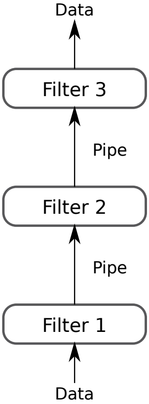
Data handling

- Data processing



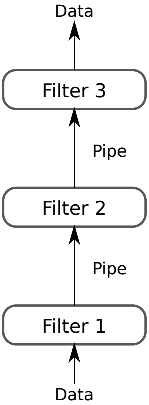
Data handling

- Data processing
- Pipes
  - Transport data



Data handling

- Data processing
- Pipes
  - Transport data
- Filters
  - Manipulate data



Popoon

```xml
<?xml version="1.0"?>

<sitemap xmlns="http://apache.org/cocoon/sitemap/1.0">

<pipelines>

  <pipeline>
    <match type="uri" pattern="examples.tgz">
      <read type="tgz" src="." name="examples.tgz"/>
    </match>
  </pipeline>

  <pipeline >
    <generate type="xmlfile" src="examples.xml"/>
    <transform type="libxslt" src="examples.xsl"/>
    <serialize type="html"/>
  </pipeline>

</pipelines>

</sitemap>
```

Popoon

```
<?xml version="1.0"?>

<sitemap xmlns="http://apache.org/cocoon/sitemap/1.0">

<pipelines>

  <pipeline>
    <match type="uri" pattern="examples.tgz">
      <read type="tgz" src="." name="examples.tgz"/>
    </match>
  </pipeline>

  <pipeline >
    <generate type="xmlfile" src="examples.xml"/>
    <transform type="libxslt" src="examples.xsl"/>
    <serialize type="html"/>
  </pipeline>

</pipelines>

</sitemap>
```

Popoon

```
<?xml version="1.0"?>

<sitemap xmlns="http://apache.org/cocoon/sitemap/1.0">

<pipelines>

  <pipeline>
    <match type="uri" pattern="examples.tgz">
      <read type="tgz" src="." name="examples.tgz"/>
    </match>
  </pipeline>

  <pipeline >
    <generate type="xmlfile" src="examples.xml"/>
    <transform type="libxslt" src="examples.xsl"/>
    <serialize type="html"/>
  </pipeline>

</pipelines>

</sitemap>
```

Popoon

```
<?xml version="1.0"?>

<sitemap xmlns="http://apache.org/cocoon/sitemap/1.0">

<pipelines>

  <pipeline>
    <match type="uri" pattern="examples.tgz">
      <read type="tgz" src="." name="examples.tgz"/>
    </match>
  </pipeline>

  <pipeline >
    <generate type="xmlfile" src="examples.xml"/>
    <transform type="libxslt" src="examples.xsl"/>
    <serialize type="html"/>
  </pipeline>

</pipelines>

</sitemap>
```

Popoon

```
<?xml version="1.0"?>

<sitemap xmlns="http://apache.org/cocoon/sitemap/1.0">

<pipelines>

  <pipeline>
    <match type="uri" pattern="examples.tgz">
      <read type="tgz" src="." name="examples.tgz"/>
    </match>
  </pipeline>

  <pipeline >
    <generate type="xmlfile" src="examples.xml"/>
    <transform type="libxslt" src="examples.xsl"/>
    <serialize type="html"/>
  </pipeline>

</pipelines>

</sitemap>
```

Pro & Contra

- Benefits:

- Drawbacks:

Pro & Contra

- Benefits:
  - Clean architectural approach
- Drawbacks:

Pro & Contra

- Benefits:
  - Clean architectural approach
  - Might gain high re-usability
- Drawbacks:

Pro & Contra

- Benefits:
  - Clean architectural approach
  - Might gain high re-usability
- Drawbacks:
  - Filters might break data easily

Pro & Contra

- Benefits:
  - Clean architectural approach
  - Might gain high re-usability
- Drawbacks:
  - Filters might break data easily
  - Somewhat forces linear code flow

Outline

Motivation

Resources

Approaches

Real world

Summary

Outline

- Real world
  - Hooks
  - Patching
  - Inheritance

Serendipity hook announcement

```php
<?php

// ... in CSS rendering code ...

// $out is CSS string
serendipity_plugin_api::hook_event('css', $out);

echo $out;
```

Serendipity hook announcement

```php
<?php

// ... in CSS rendering code ...

// $out is CSS string
serendipity_plugin_api::hook_event('css', $out);

echo $out;
```

Serendipity hook reaction

```
class serendipity_event_bbcode extends serendipity_event
{
    function event_hook($event, &$bag, &$eventData) {
        switch($event) {
            case 'css':
                if (strpos($eventData, '.bb-code') !== false) {
                    // class exists in CSS ...
                    return true;
                }
    ?>
    .bb-quote, .bb-code, .bb-php, .bb-code-title, .bb-php-title {
        margin-left: 20px;
        margin-right: 20px;
        /* ... */
    }
    /* ... */
    <?php
            return true;
            break;

            default:
              return false;
        }
    }
}
```

Serendipity hook reaction

```
class serendipity_event_bbcode extends serendipity_event
{
    function event_hook($event, &$bag, &$eventData) {
        switch($event) {
            case 'css':
                if (strpos($eventData, '.bb-code') !== false) {
                    // class exists in CSS ...
                    return true;
                }
    ?>
    .bb-quote, .bb-code, .bb-php, .bb-code-title, .bb-php-title {
        margin-left: 20px;
        margin-right: 20px;
        /* ... */
    }
    /* ... */
    <?php
            return true;
            break;

            default:
              return false;
        }
    }
}
```

Serendipity hook reaction

```
class serendipity_event_bbcode extends serendipity_event
{
    function event_hook($event, &$bag, &$eventData) {
        switch($event) {
            case 'css':
                if (strpos($eventData, '.bb-code') !== false) {
                    // class exists in CSS ...
                    return true;
                }
    ?>
    .bb-quote, .bb-code, .bb-php, .bb-code-title, .bb-php-title {
        margin-left: 20px;
        margin-right: 20px;
        /* ... */
    }
    /* ... */
    <?php
            return true;
            break;

            default:
              return false;
        }
    }
}
```

Serendipity hook reaction

```
class serendipity_event_bbcode extends serendipity_event
{
    function event_hook($event, &$bag, &$eventData) {
        switch($event) {
            case 'css':
                if (strpos($eventData, '.bb-code') !== false) {
                    // class exists in CSS ...
                    return true;
                }
    ?>
    .bb-quote, .bb-code, .bb-php, .bb-code-title, .bb-php-title {
        margin-left: 20px;
        margin-right: 20px;
        /* ... */
    }
    /* ... */
    <?php
            return true;
            break;

            default:
              return false;
        }
    }
}
```

Serendipity hook reaction

```
class serendipity_event_bbcode extends serendipity_event
{
    function event_hook($event, &$bag, &$eventData) {
        switch($event) {
            case 'css':
                if (strpos($eventData, '.bb-code') !== false) {
                    // class exists in CSS ...
                    return true;
                }
    ?>
    .bb-quote, .bb-code, .bb-php, .bb-code-title, .bb-php-title {
        margin-left: 20px;
        margin-right: 20px;
        /* ... */
    }
    /* ... */
    <?php
            return true;
            break;

            default:
              return false;
        }
    }
}
```

Serendipity hook reaction

```
class serendipity_event_bbcode extends serendipity_event
{
    function event_hook($event, &$bag, &$eventData) {
        switch($event) {
            case 'css':
                if (strpos($eventData, '.bb-code') !== false) {
                    // class exists in CSS ...
                    return true;
                }
    ?>
    .bb-quote, .bb-code, .bb-php, .bb-code-title, .bb-php-title {
        margin-left: 20px;
        margin-right: 20px;
        /* ... */
    }
    /* ... */
    <?php
            return true;
            break;

            default:
              return false;
        }
    }
}
```

Pro & Contra

- Benefits:

- Drawbacks:

Pro & Contra

- Benefits:
  - High flexibility
- Drawbacks:

Pro & Contra

- Benefits:
  - High flexibility
  - Low coding efforts
- Drawbacks:

Pro & Contra

- Benefits:
  - High flexibility
  - Low coding efforts
- Drawbacks:
  - Plugin can easily break hook data

Pro & Contra

- Benefits:
  - High flexibility
  - Low coding efforts
- Drawbacks:
  - Plugin can easily break hook data
  - No defined data formats

Pro & Contra

- Benefits:
  - High flexibility
  - Low coding efforts
- Drawbacks:
  - Plugin can easily break hook data
  - No defined data formats
  - “Liskov substitution principle” limits what you are allowed to do

Outline

Real world

- Hooks
- Patching
- Inheritance

Patching the source

- The naive approach

Patching the source

- The naive approach
- Works suprisingly well
  - For some of the largest module ecosystems
  - phpBB

phpBB MODx format

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<!-- ... -->
<mod xmlns="http://www.phpbb.com/mods/xml/modx-1.2.0.xsd">
    <header><!-- ... --></header>
    <!-- ... -->
        <open src="index.php">
            <edit>
                <comment lang="en">Here is a comment</comment>
                <comment lang="nl">Hier is een stukje commentaar</comment>
                <find>text to find</find>
                <action type="replace-with">text to be replaced with</action>
            </edit>
            <edit>
                <find>text to find</find>
                <action type="after-add">text to be added on the line after</action>
            </edit>
            <!-- ... -->
        </open>
    <!-- ... -->
</mod>
```

phpBB MODx format

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<!-- ... -->
<mod xmlns="http://www.phpbb.com/mods/xml/modx-1.2.0.xsd">
    <header><!-- ... --></header>
    <!-- ... -->
        <open src="index.php">
            <edit>
                <comment lang="en">Here is a comment</comment>
                <comment lang="nl">Hier is een stukje commentaar</comment>
                <find>text to find</find>
                <action type="replace-with">text to be replaced with</action>
            </edit>
            <edit>
                <find>text to find</find>
                <action type="after-add">text to be added on the line after</action>
            </edit>
            <!-- ... -->
        </open>
    <!-- ... -->
</mod>
```

phpBB MODx format

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<!-- ... -->
<mod xmlns="http://www.phpbb.com/mods/xml/modx-1.2.0.xsd">
    <header><!-- ... --></header>
    <!-- ... -->
        <open src="index.php">
            <edit>
                <comment lang="en">Here is a comment</comment>
                <comment lang="nl">Hier is een stukje commentaar</comment>
                <find>text to find</find>
                <action type="replace-with">text to be replaced with</action>
            </edit>
            <edit>
                <find>text to find</find>
                <action type="after-add">text to be added on the line after</action>
            </edit>
            <!-- ... -->
        </open>
    <!-- ... -->
</mod>
```

phpBB MODx format

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<!-- ... -->
<mod xmlns="http://www.phpbb.com/mods/xml/modx-1.2.0.xsd">
    <header><!-- ... --></header>
    <!-- ... -->
        <open src="index.php">
            <edit>
                <comment lang="en">Here is a comment</comment>
                <comment lang="nl">Hier is een stukje commentaar</comment>
                <find>text to find</find>
                <action type="replace-with">text to be replaced with</action>
            </edit>
            <edit>
                <find>text to find</find>
                <action type="after-add">text to be added on the line after</action>
            </edit>
            <!-- ... -->
        </open>
    <!-- ... -->
</mod>
```

Pro & Contra

- Benefits:

- Drawbacks:

Pro & Contra

- Benefits:
  - Trivial to get started with (high “hackability”)
- Drawbacks:

Pro & Contra

- Benefits:
  - Trivial to get started with (high “hackability”)
  - You can change anything
- Drawbacks:

Pro & Contra

- Benefits:
  - Trivial to get started with (high “hackability”)
  - You can change anything
- Drawbacks:
  - Will definitely break

Pro & Contra

- Benefits:
  - Trivial to get started with (high “hackability”)
  - You can change anything
- Drawbacks:
  - Will definitely break
  - Can lead to unparsable code

Pro & Contra

- Benefits:
  - Trivial to get started with (high “hackability”)
  - You can change anything
- Drawbacks:
  - Will definitely break
  - Can lead to unparsable code
  - Complex modules require deep knowledge

Outline

Real world

- Hooks
- Patching
- Inheritance

Inheritance

- Generally:
  - Use Aggregation for code re-use
  - Not inheritance!

Inheritance

- Generally:
  - Use Aggregation for code re-use
  - Not inheritance!
- Oxid eSales (OS shop software)
  - Interesting extension model
  - Built entirely on inheritance

Inheritance

- Generally:
  - Use Aggregation for code re-use
  - Not inheritance!
- Oxid eSales (OS shop software)
  - Interesting extension model
  - Built entirely on inheritance
  - Each module can inherit from “any” class

Inheritance

- Generally:
  - Use Aggregation for code re-use
  - Not inheritance!
- Oxid eSales (OS shop software)
  - Interesting extension model
  - Built entirely on inheritance
  - Each module can inherit from “any” class
  - Each inheriting class will be used everywhere

Inheritance

- Generally:
  - Use Aggregation for code re-use
  - Not inheritance!
- Oxid eSales (OS shop software)
  - Interesting extension model
  - Built entirely on inheritance
  - Each module can inherit from “any” class
  - Each inheriting class will be used everywhere
  - *How can that be possible?*

Modular inheritance

- `oxnew("ClassName")` instead of `new ClassName`
- Inheritance graph created on-the-fly
- Generates intermediate classes

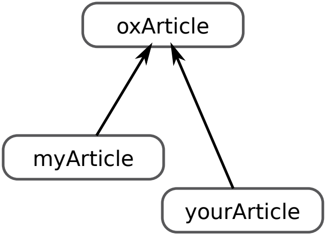
Example

```php
<?php

class oxArticle
{
    public function calculatePrice()
    {
        // ...
    }
}

class myArticle
    extends oxArticle
{
    // ...
}

class yourArticle
    extends oxArticle
{
    // ...
}
```



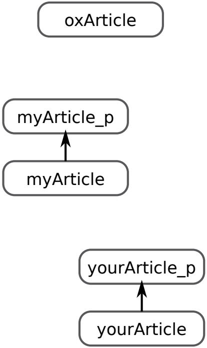
Example

```php
<?php

class oxArticle
{
    public function calculatePrice()
    {
        // ...
    }
}

class myArticle
    extends myArticle_parent
{
    // ...
}

class yourArticle
    extends yourArticle_parent
{
    // ...
}
```



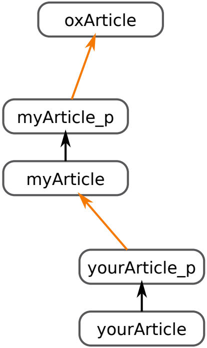
Example

```php
<?php

class oxArticle
{
    public function calculatePrice()
    {
        // ...
    }
}

class myArticle
    extends myArticle_parent
{
    // ...
}

class yourArticle
    extends yourArticle_parent
{
    // ...
}
```



Pro & Contra

- Benefits:
- Drawbacks:

Pro & Contra

- Benefits:
  - You can extend about everything…
- Drawbacks:

Pro & Contra

- Benefits:
  - You can extend about everything…
- Drawbacks:
  - About everything will be extended…

Pro & Contra

- Benefits:
  - You can extend about everything...
- Drawbacks:
  - About everything will be extended...
  - You may not use the `new` operator

Pro & Contra

- Benefits:
  - You can extend about everything…
- Drawbacks:
  - About everything will be extended…
  - You may not use the `new` operator
  - Violates object-oriented design principles

Pro & Contra

- Benefits:
  - You can extend about everything. . .
- Drawbacks:
  - About everything will be extended. . .
  - You may not use the `new` operator
  - Violates object-oriented design principles
  - Non-enforcable constraints (`parent::method()`)

Pro & Contra

- Benefits:
  - You can extend about everything…
- Drawbacks:
  - About everything will be extended…
  - You may not use the `new` operator
  - Violates object-oriented design principles
  - Non-enforcable constraints (`parent::method()`)
  - Almost untestable

Pro & Contra

- Benefits:
  - You can extend about everything…
- Drawbacks:
  - About everything will be extended…
  - You may not use the `new` operator
  - Violates object-oriented design principles
  - Non-enforcable constraints (`parent::method()`)
  - Almost untestable
  - *Everything* becomes public API

Outline

Motivation

Resources

Approaches

Real world

Summary

Summary

- Patching
- Hooks
- Pipes & Filters
- Inheritance
- Subject-Observer
- Signal-Slot

Thanks for Listening

Questions? Comments? Critics? Ideas?

Please rate this talk at
`https://joind.in/6102`

Thanks for Listening

Please rate this talk at
`https://joind.in/6102`
(Slides will be linked there)

Stay in touch

- Tobias Schlitt
- toby@qafoo.com
- @tobySen / @qafoo

Rent a PHP quality expert:
`http://qafoo.com`