
Modular Application Architecture

International PHP Conference – Spring Edition 2011

Kore Nordmann (@koredn)
Tobias Schlitt (@tobySen)

May 29, 2011

About us

- ▶ Degree in computer science



About us

- ▶ Degree in computer science
- ▶ More than 10 years of professional PHP



About us

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



About us

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

Co-founders of



About us

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

Co-founders of



**We help people to create
high quality PHP
applications.**

About us

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

Co-founders of



**We help people to create
high quality PHP
applications.**

<http://qafoo.com>

Outline

Motivation

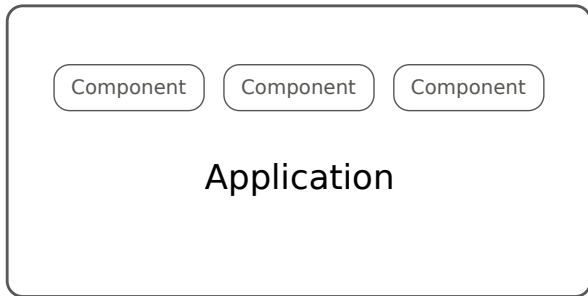
Resources

Approaches

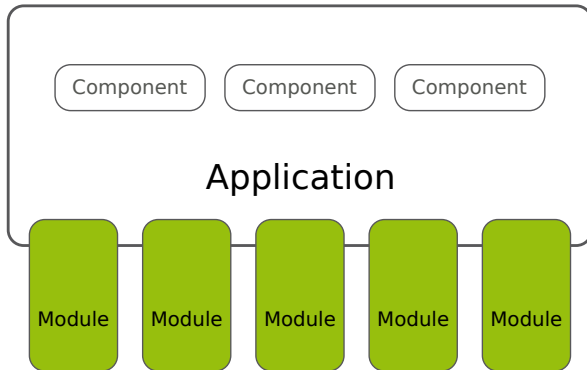
Real world

Summary

Application



Modules



Why modules?

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

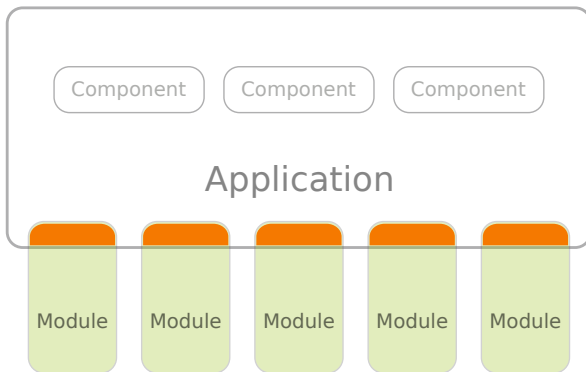
Why modules?

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

Why modules?

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

Essential



Challenges

- ▶ Module structure
- ▶ Registration / configuration
- ▶ Handling resources
- ▶ Interaction with core

Challenges

- ▶ Module structure ✓
- ▶ Registration / configuration
- ▶ Handling resources
- ▶ Interaction with core

Challenges

- ▶ Module structure ✓
- ▶ Registration / configuration ✓
- ▶ Handling resources
- ▶ Interaction with core

Challenges

- ▶ Module structure ✓
- ▶ Registration / configuration ✓
- ▶ Handling resources →
- ▶ Interaction with core

Challenges

- ▶ Module structure ✓
- ▶ Registration / configuration ✓
- ▶ Handling resources →
- ▶ 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

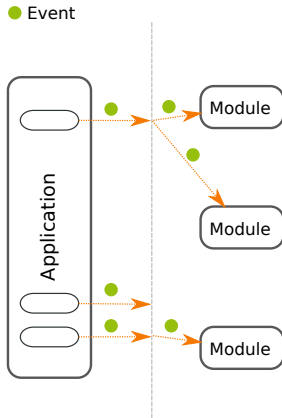
Approaches

Event handling

Data handling

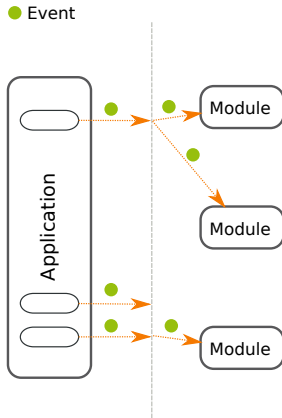
Event handling

► Interaction



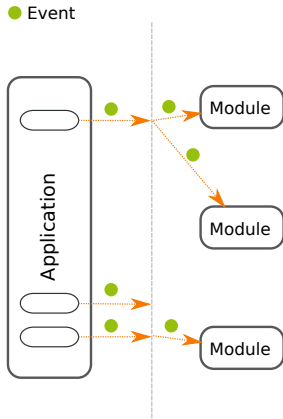
Event handling

- ▶ Interaction
- ▶ Modules register for event types



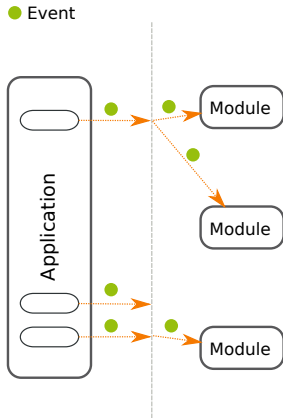
Event handling

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



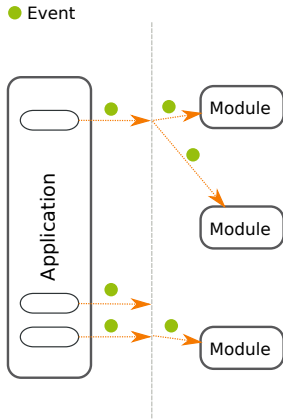
Event handling

- ▶ Interaction
- ▶ Modules register for event types
- ▶ Events “thrown” (by core or module)
- ▶ Registered modules informed about event (maybe including data)



Event handling

- ▶ Interaction
- ▶ Modules register for event types
- ▶ Events “thrown” (by core or module)
- ▶ Registered modules informed about event (maybe including data)
- ▶ Optionally transparent



Subject-Observer

```
1 <?php
2
3 class Subject
4 {
5     public function doSomething()
6     {
7         $this->notify( 'doSomethingStart' );
8         // ...
9         $this->notify( 'doSomethingEnd' );
10    }
11 }
```

Subject-Observer

```
1 <?php
2
3 class Subject
4 {
5     protected $observers = array();
6
7     public function addObserver( Observer $observer )
8     {
9         $this->observers [] = $observer;
10    }
11
12    public function notify( $event, $data = null )
13    {
14        foreach ( $this->observers as $observer )
15        {
16            $observer->$event( $data );
17        }
18    }
19
20    public function doSomething()
21    {
22        $this->notify( 'doSomethingStart' );
23        // ...
24        $this->notify( 'doSomethingEnd' );
25    }
26 }
```


Subject-Observer

```
1 <?php
3 class Observer
4 {
5     public function doSomethingStart ()
6     {
7         // ...
8     }
9
10    public function doSomethingEnd ()
11    {
12        // ...
13    }
14 }
```

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 signal 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 signal 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 signal will take to process
 - ▶ Limited to defined extension points
 - ▶ **Requires implementation in each subject**

Signal slot

```
1 <?php
2
3 $handler = new arbitSignalSlot();
4
5 $handler->register( 'signalA', array( new myModule(), 'handleSignalA' ) );
6 $handler->register( 'signalA', array( new yourModule(), 'handleSignalA' ) );
7
8 // In module c
9 $handler->emit( 'signalA', new signalADataStruct( /* ... */ ) );
10
11 // Now all modules registered for this signal are called with the provided data
12 class myModule
13 {
14     public function handleSignalA( $name, signalADataStruct $data )
15     {
16         // ...
17     }
18 }
```

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

Outline

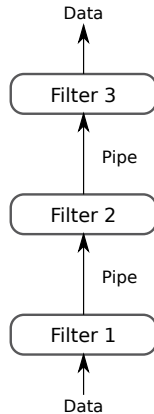
Approaches

Event handling

Data handling

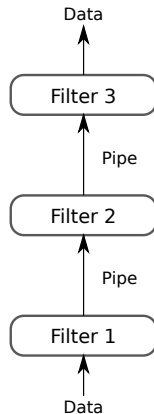
Data handling

- ▶ Data processing



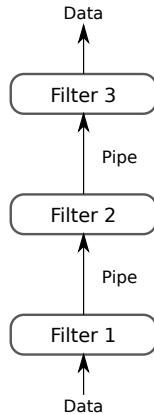
Data handling

- ▶ Data processing
- ▶ Pipes
 - ▶ Transport data



Data handling

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



Popoon

```
1 <?xml version="1.0" ?>
2
3 <map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
4 <map:pipelines>
5   <map:pipeline>
6     <map:match type="uri" pattern="examples.tgz">
7       <map:read type="tgz" src="." name="examples.tgz" />
8     </map:match>
9   </map:pipeline>
10
11   <map:pipeline >
12     <map:generate type="xmlfile" src="examples.xml" />
13     <map:transform type="libxslt" src="examples.xsl" />
14     <map:serialize type="html" />
15   </map:pipeline>
16 </map:pipelines>
17 </map:sitemap>
```

Popoon

```
1 <?xml version="1.0" ?>
2
3 <map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
4 <map:pipelines>
5   <map:pipeline>
6     <map:match type="uri" pattern="examples.tgz">
7       <map:read type="tgz" src="." name="examples.tgz" />
8     </map:match>
9   </map:pipeline>
10
11   <map:pipeline >
12     <map:generate type="xmlfile" src="examples.xml" />
13     <map:transform type="libxslt" src="examples.xsl" />
14     <map:serialize type="html" />
15   </map:pipeline>
16 </map:pipelines>
17 </map:sitemap>
```

Popoon

```
1 <?xml version="1.0" ?>
2
3 <map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
4 <map:pipelines>
5   <map:pipeline>
6     <map:match type="uri" pattern="examples.tgz">
7       <map:read type="tgz" src="." name="examples.tgz" />
8     </map:match>
9   </map:pipeline>
10
11   <map:pipeline >
12     <map:generate type="xmlfile" src="examples.xml" />
13     <map:transform type="libxslt" src="examples.xsl" />
14     <map:serialize type="html" />
15   </map:pipeline>
16 </map:pipelines>
17 </map:sitemap>
```

Popoon

```
1 <?xml version="1.0" ?>
2
3 <map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
4 <map:pipelines>
5   <map:pipeline>
6     <map:match type="uri" pattern="examples.tgz">
7       <map:read type="tgz" src="." name="examples.tgz" />
8     </map:match>
9   </map:pipeline>
10
11   <map:pipeline >
12     <map:generate type="xmlfile" src="examples.xml" />
13     <map:transform type="libxslt" src="examples.xsl" />
14     <map:serialize type="html" />
15   </map:pipeline>
16 </map:pipelines>
17 </map:sitemap>
```


Popoon

```
1 <?xml version="1.0" ?>
2
3 <map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
4 <map:pipelines>
5   <map:pipeline>
6     <map:match type="uri" pattern="examples.tgz">
7       <map:read type="tgz" src="." name="examples.tgz" />
8     </map:match>
9   </map:pipeline>
10
11   <map:pipeline >
12     <map:generate type="xmlfile" src="examples.xml" />
13     <map:transform type="libxslt" src="examples.xsl" />
14     <map:serialize type="html" />
15   </map:pipeline>
16 </map:pipelines>
17 </map: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

```
1 <?php
2
3 // ... in CSS code ...
4
5 // $out is CSS string
6 serendipity_plugin_api::hook_event($css_hook, $out);
7
8 echo $out;
9
10 // ... in entry display code ...
11
12 // $entry is blog entry
13 // $addData is meta data
14 serendipity_plugin_api::hook_event('frontend_display', $entry, $addData);
```

Serendipity hook announcement

```
1 <?php
2
3 // ... in CSS code ...
4
5 // $out is CSS string
6 serendipity_plugin_api::hook_event($css_hook , $out);
7
8 echo $out;
9
10 // ... in entry display code ...
11
12 // $entry is blog entry
13 // $addData is meta data
14 serendipity_plugin_api::hook_event('frontend_display' , $entry , $addData);
```

Serendipity hook reaction

```
18 function event_hook($event, &$amp;bag, &$amp;eventData) {
19     global $serendipity;
20
21     $hooks = &$amp;bag->get('event_hooks');
22
23     if (isset($hooks[$event])) {
24         switch($event) {
25             case 'frontend_display':
26                 if ( $condition /* ... */ ) {
27                     $element = $temp['element'];
28                     $eventData[$element] = $this->bbcode(
29                         $eventData[$element]
30                     );
31                 }
32                 return true;
33                 break;

```

Serendipity hook reaction

```
18 function event_hook($event, &$amp;bag, &$amp;eventData) {
19     global $serendipity;
20
21     $hooks = &$amp;bag->get('event_hooks');
22
23     if (isset($hooks[$event])) {
24         switch($event) {
25             case 'frontend_display':
26                 if ( $condition /* ... */ ) {
27                     $element = $temp['element'];
28                     $eventData[$element] = $this->bbcode(
29                         $eventData[$element]
30                     );
31                 }
32                 return true;
33                 break;

```

Serendipity hook reaction

```
18 function event_hook($event, &$amp;bag, &$amp;eventData) {
19     global $serendipity;
20
21     $hooks = &$amp;bag->get('event_hooks');
22
23     if (isset($hooks[$event])) {
24         switch($event) {
25             case 'frontend_display':
26                 if ( $condition /* ... */ ) {
27                     $element = $temp['element'];
28                     $eventData[$element] = $this->bbcode(
29                         $eventData[$element]
30                     );
31                 }
32                 return true;
33                 break;

```

Serendipity hook reaction

```
18 function event_hook($event, &$bag, &$eventData) {
19     global $serendipity;
20
21     $hooks = &$bag->get('event_hooks');
22
23     if (isset($hooks[$event])) {
24         switch($event) {
25
26             ...
27
28             case 'css':
29                 if (strpos($eventData, '.bb-code') !== false) {
30                     // class exists in CSS, so a user has customized it and
31                     // we don't need default
32                     return true;
33                 }
34             ?>
35             .bb-quote, .bb-code, .bb-php, .bb-code-title, .bb-php-title {
36                 margin-left: 20px;
37                 margin-right: 20px;
38                 /* ... */
39             }
40             /* ... */
41             <?php
42                 return true;
43                 break;
44
45         }
46     }
47 }
48
49
```

Serendipity hook reaction

```
18 function event_hook($event, &$amp;bag, &$amp;eventData) {
19     global $serendipity;
20
21     $hooks = &$amp;bag->get('event_hooks');
22
23     if (isset($hooks[$event])) {
24         switch($event) {
25
26             ...
27
28             case 'css':
29                 if (strpos($eventData, '.bb-code') !== false) {
30                     // class exists in CSS, so a user has customized it and
31                     // we don't need default
32                     return true;
33                 }
34             ?>
35             .bb-quote, .bb-code, .bb-php, .bb-code-title, .bb-php-title {
36                 margin-left: 20px;
37                 margin-right: 20px;
38                 /* ... */
39             }
40             /* ... */
41             <?php
42                 return true;
43                 break;
44
45         }
46     }
47 }
48
49
```

Serendipity hook reaction

```
18 function event_hook($event, &$amp;bag, &$amp;eventData) {
19     global $serendipity;
20
21     $hooks = &$amp;bag->get('event_hooks');
22
23     if (isset($hooks[$event])) {
24         switch($event) {
25
26             ...
27
28             case 'css':
29                 if (strpos($eventData, '.bb-code') !== false) {
30                     // class exists in CSS, so a user has customized it and
31                     // we don't need default
32                     return true;
33                 }
34
35                 ?>
36                 .bb-quote, .bb-code, .bb-php, .bb-code-title, .bb-php-title {
37                     margin-left: 20px;
38                     margin-right: 20px;
39                     /* ... */
40                 }
41                 /* ... */
42                 <?php
43                 return true;
44                 break;
45
46             ...
47
48             ...
49         }
50     }
51 }
```


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 surprisingly well for some of the largest module ecosystems: phpBB

phpBB MODx format

```
1 <?xml version="1.0" encoding="utf-8" standalone="yes" ?>
2 <?xml-stylesheet type="text/xsl" href="1.2.0/modx.prosilver.en.xsl"?>
3 <mod xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.
  phpbb.com/mods/xml/modx-1.2.0.xsd">
4   <header>
42  </header>
43   <open src="index.php">
44     <edit>
45       <comment lang="en">Here is a comment</comment>
46       <comment lang="nl">Hier is een stukje commentaar</comment>
47       <find>text to find</find>
48       <action type="replace-with">text to be replaced with</action>
49     </edit>
50     <edit>
51       <find>text to find</find>
52       <action type="after-add">text to be added on the line after</
  action>
53     </edit>
54     <edit>
55       <find>text to find</find>
56       <action type="before-add">text to be added on the line before</
  action>
57   </edit>
```

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 unparseable 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 instead of inheritance for code re-use.

Inheritance

- ▶ Generally: Use Aggregation instead of inheritance for code re-use.
- ▶ Oxid eSales (OS shop software) has an interesting extension model build entirely on inheritance
 - ▶ Any number of modules can inherit from “any” class ...

Inheritance

- ▶ Generally: Use Aggregation instead of inheritance for code re-use.
- ▶ Oxid eSales (OS shop software) has an interesting extension model build entirely on inheritance
 - ▶ Any number of modules can inherit from “any” class ...
 - ▶ ... and each inheriting class will be used anywhere the original object would be used.

Inheritance

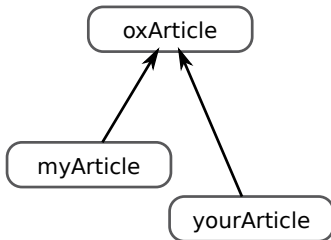
- ▶ Generally: Use Aggregation instead of inheritance for code re-use.
- ▶ Oxid eSales (OS shop software) has an interesting extension model build entirely on inheritance
 - ▶ Any number of modules can inherit from “any” class ...
 - ▶ ... and each inheriting class will be used anywhere the original object would be used.
 - ▶ *How can that be possible?*

Modular inheritance

- ▶ Objects are instantiated with a special function instead of the `new` operator.
- ▶ Inheritance graph is created on-the-fly by generating intermediate classes

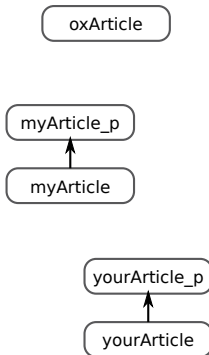
Example

```
1 <?php
2
3 class oxArticle
4 {
5     public function calculatePrice()
6     {
7         // ...
8     }
9 }
10
11 class myArticle
12     extends oxArticle
13 {
14     // ...
15 }
16
17 class yourArticle
18     extends oxArticle
19 {
20     // ...
21 }
```



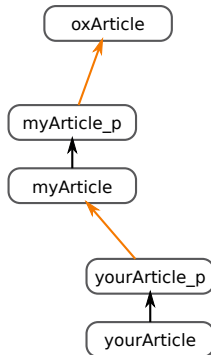
Example

```
1 <?php
2
3 class oxArticle
4 {
5     public function calculatePrice()
6     {
7         // ...
8     }
9 }
10
11 class myArticle
12     extends myArticle_parent
13 {
14     // ...
15 }
16
17 class yourArticle
18     extends yourArticle_parent
19 {
20     // ...
21 }
```



Example

```
1 <?php
2
3 class oxArticle
4 {
5     public function calculatePrice()
6     {
7         // ...
8     }
9 }
10
11 class myArticle
12     extends myArticle_parent
13 {
14     // ...
15 }
16
17 class yourArticle
18     extends yourArticle_parent
19 {
20     // ...
21 }
```



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 – but use something like:
`oxNew("oxArticle")`

Pro & Contra

- ▶ **Benefits:**
 - ▶ You can extend about everything...
- ▶ **Drawbacks:**
 - ▶ About everything will be extended...
 - ▶ You may not use the `new` operator – but use something like:
`oxNew("oxArticle")`
 - ▶ 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 – but use something like:
`oxNew("oxArticle")`
 - ▶ 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 – but use something like:
`oxNew("oxArticle")`
 - ▶ Violates object-oriented design principles
 - ▶ Non-enforcable constraints (`parent::method()`)
 - ▶ Liskov substitution principle limits what you are allowed to do

Outline

Motivation

Resources

Approaches

Real world

Summary

Summary

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

Thanks for listening

Please rate this talk at
<http://joind.in/3484>

Slides will be online (soon)
<http://talks.qafoo.com>

Thanks for listening

Please rate this talk at
<http://joind.in/3484>

Stay in touch

- ▶ Kore Nordmann
- ▶ kore@qafoo.com
- ▶ @koredn / @qafoo
- ▶ Tobias Schlitt
- ▶ toby@qafoo.com
- ▶ @tobySen / @qafoo

Rent a PHP quality expert:
<http://qafoo.com>