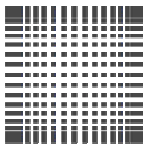
Masterseminar Informatik

Aspektorientierte Programmierung mit



hochschule mannheim

Ralf Haingärtner / Markus Pech

04.05.2006

Agenda

Übersicht

- Was ist PHP?
- Funktionsweise
- Besonderheiten
- AOP mit PHP
- aopinphp
- Bunny Aspects
- Seasar
- aoPHP
- MFAOPHP
- phpaspect
- Fazit

- Was ist PHP
- Funktionsweise von PHP
- Besonderheiten von PHP
- Allgemeines zu AOP mit PHP
- AOP Ansätze
  - aopinphp
  - Bunny Aspects
  - Seasar
  - aoPHP
  - MFAOPHP
  - phpaspect
- Fazit

Ralf Haingärtner / Markus Pech
Hochschule Mannheim

Was ist PHP ? 1/4

„**P**HP: **H**ypertext **P**reprocessor“

früher

„**P**ersonal **H**ome **P**age Tools“

- PHP ist eine Skriptsprache
  - Weniger mächtig als “echte” Programmiersprachen
  - Vorteilhaft bei der schnellen Erstellung von kleinen Programmen
  - Nachteilig bei größeren Projekten
- PHP ist Webtechnologie
  - Dient hauptsächlich der Erstellung dynamischer Websiten...
  - oder ganzer Webanwendungen
  - Auch für Offline-Skripte benutztbar
- PHP wird serverseitig interpretiert
  - Keine Anforderungen an den Client (Browser)
  - Kein Quellcode einsehbar
  - Ausgabe von HTML, PDF, Bildern etc.

Was ist PHP ? 2/4

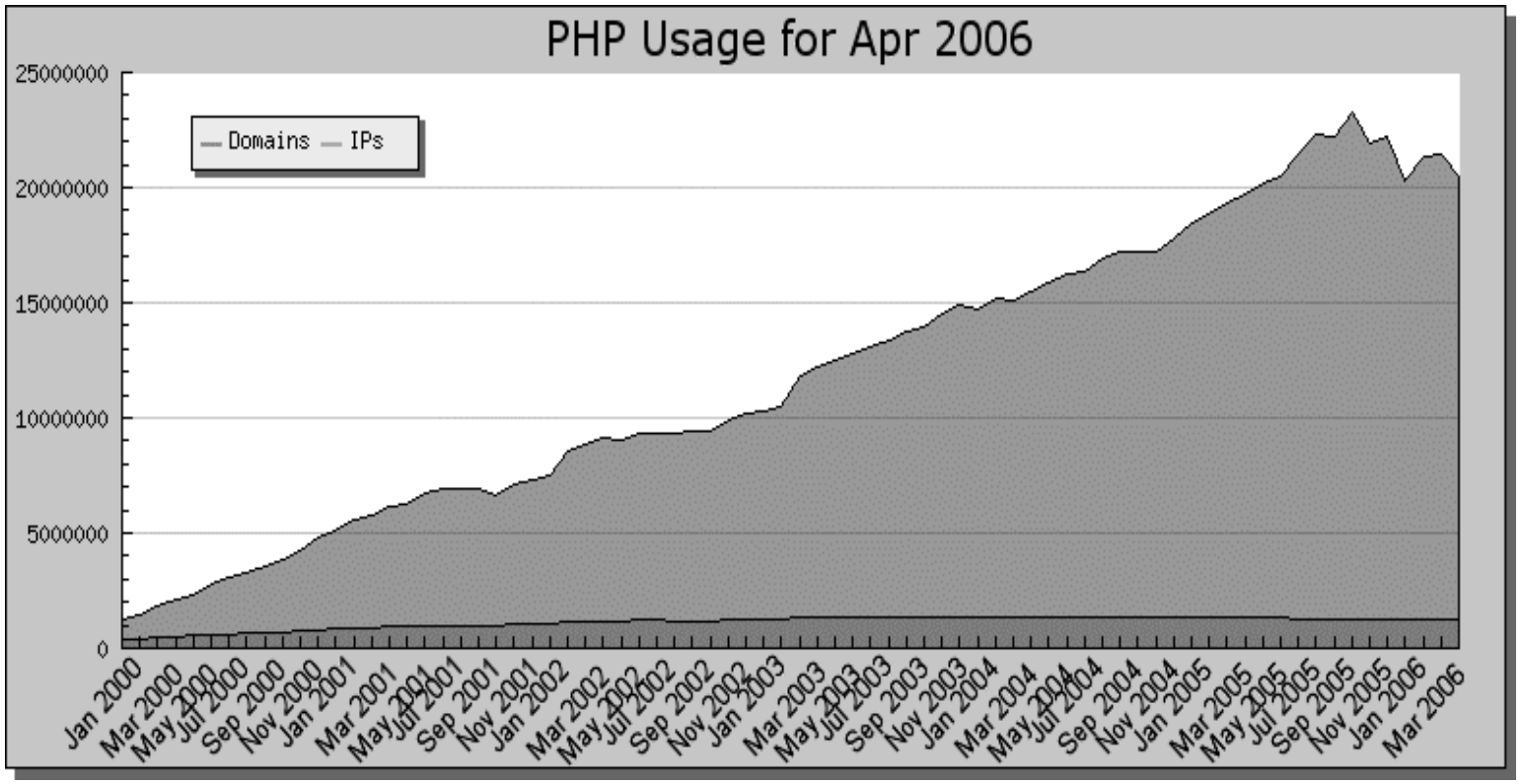
- PHP ist leicht erlernbar
  - Überschaubarer Sprachumfang
  - Einfache Sprachkonstrukte
  - Umfangreiche Dokumentation / viele Beispiele, Tutorials
- PHP ist Open Source
  - Große Entwicklergemeinde
  - Stetige Weiterentwicklung
  - Viele aktive Projekte
- PHP hat viele Funktionsbibliotheken
  - Umfangreiche Spracherweiterungen
  - Anbindung an andere Technologien

Was ist PHP ? 3/4

- PHP 1.0 erschienen Juni 1995
  - Sammlung von Perl-Skripten
- PHP 2.0.0 bzw. PHP/FI, erschienen November 1997
- PHP 3.0.0, erschienen Juni 1998
- PHP 4.0.0, erschienen Mai 2000
  - Einfache Objektorientierte Programmierung
  - Zend Engine 1
- PHP 5.0.0, erschienen Juli 2004
  - Objektorientiertes Programmieren
  - Zend Engine 2
- PHP 5.1.0, erschienen November 2005
- PHP 6.0.0 angekündigt für das dritte Quartal 2006
  - umfassende Unicode-Unterstützung

Was ist PHP ? 4/4

Verbreitung von PHP



Funktionsweise von PHP 1/3

1. Ein Client (Browser) startet eine Anfrage für eine PHP-Datei per URL
2. Der Server nimmt die Anfrage entgegen und lädt die Datei
3. Der Server übergibt die Datei an den PHP-Interpreter
4. Der PHP-Interpreter arbeitet das PHP-Skript ab
5. Der PHP-Interpreter gibt das Ergebnis an den Server zurück
6. Der Server liefert die Antwort an den Client

Funktionsweise von PHP 2/3

Codebeispiel:

```
<?php echo "Hello World!"; ?>
```

Code eingebettet in HTML:

```
<html>
        <head><title> Hello World!-Beispiel</title></head>
        <body>
                <?php echo "Hello World!"; ?>
        </body>
</html>
```

Funktionsweise von PHP 3/3

- include() / require()
  - Eingebundener Code erbt den Geltungsbereich von Variablen der Zeile in der er eingebunden wird
  - Variablen und Funktionen des eingebundenen Codes gelten ab dieser Zeile im aufrufenden Skript
- Beispiel:

vars.php

```
<?php
    $color = 'grün';
    $fruit = 'Apfel';
?>
```

test.php

```
<?php
    echo "Ein $color $fruit"; // Ein
    include 'vars.php';
    echo "Ein $color $fruit"; // Ein grüner Apfel
?>
```

Besonderheiten von PHP 1/6

- Keine Persistenz
- Skripte arbeiten nur bis zur Fertigstellung der Ausgabe
- Variablen und Objekte verlieren mit dem Ende der Ausführung ihres Skripts die Gültigkeit
- Sicherung von Variablen und Objekten über Serialisierung in einem Session-Array
- PHP-Applikationen bestehen aus Einzelskripten, Kommunikation per gegenseitigem Aufruf oder Einbinden
- Übermittlung von Parametern über GET- bzw. POST-Array

Besonderheiten von PHP 2/6

- Meist Fremdhosting
  - kein Zugriff auf Serverkonfiguration
  - Einbinden von Erweiterungen schwierig
- LAMP/WAMP-Pakete
  - Linux / Windows
  - Apache Webserver
  - MySql
  - PHP
- Prozedurale Programmierung
  - Einfache Skripte
  - Persöniche Webseiten
  - Kleine Projekte

Besonderheiten von PHP 3/6

Übersicht

- Objektmodell PHP 4
  - Großteils prozedural angelegt
  - Keine Übergabe von Referenzen bei Zuweisungen
  - Keine Destruktoren
  - Keine Sichtbarkeitsmodifikatoren
  - Keine Kapselung
  - Verlust des Klassenbezugs beim deserialisieren von Objekten möglich

Besonderheiten von PHP 4/6

Übersicht

- Objektmodell PHP 5
  - Sichtbarkeiten von Variablen und Methoden
  - Objektreferenzen statt Kopien
  - Destruktoren
  - Klonen von Objekten
  - “Magische” __Funktionen
  - Interfaces
  - Final / Abstract
  - Exceptions
  - Type Hinting

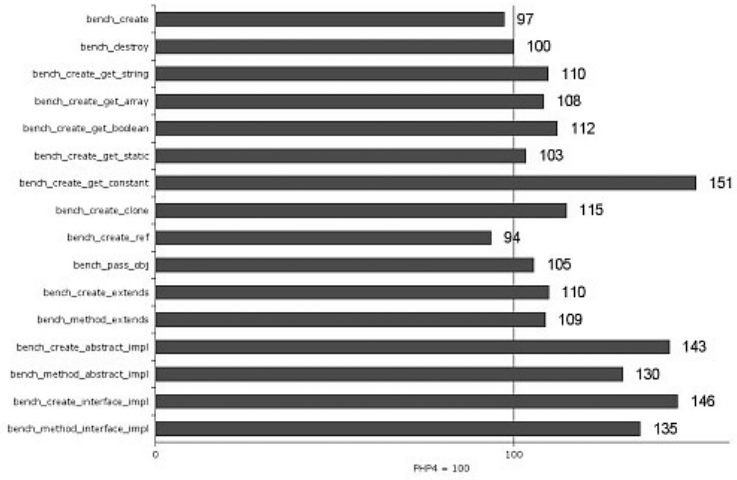
Besonderheiten von PHP 5/6

- Beispiel: Type Hinting

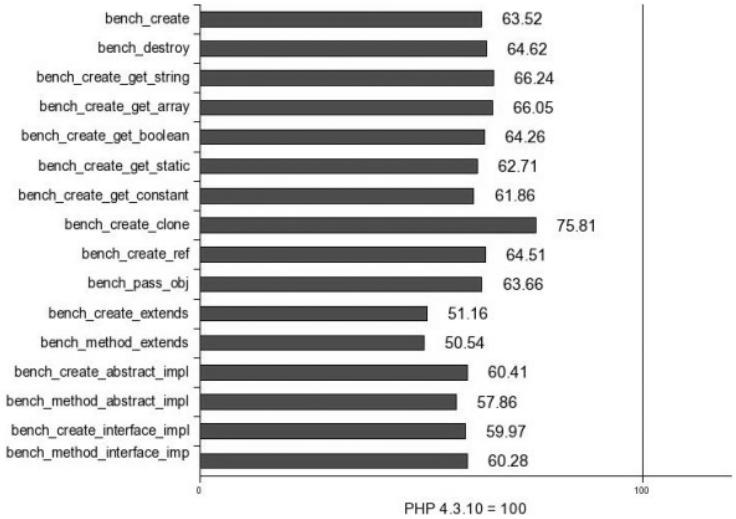
```
<?php
   class MyClass
   {
      // Der erste Parameter muss ein Objekt des Typs OtherClass sein
      public function test(OtherClass $otherclass) {
          echo $otherclass->var;
      }

      // Der erste Parameter muss ein Array sein
      public function test_array(array $input_array) {
          print_r($input_array);
      }
   }
?>
```

PHP 5.1

PHP 5.1 ist ein inkrementelles Update auf der Basis der Zend II Engine und des Objektmodells von PHP 5. Die Änderungen sind jedoch - auch wenn sie nur Module betreffen - signifikant.



PHP 5.0.0



PHP 5.1.0

Fragen

Übersicht

- Was ist PHP?
- Funktionsweise
- Besonderheiten
- AOP mit PHP
- aopinphp
- Bunny Aspects
- Seasar
- aoPHP
- MFAOPHP
- phpaspect
- Fazit

AOP mit PHP 1/3

- Erst seit kurzem vernünftige Umsetzung der Objektorientierung
- In der praktischen Anwendung derzeit Übergang von prozeduraler Programmierung zu OOP
- Großer Anteil von „Hobbyentwicklern“ im PHP-Umfeld
  - Kleinere Projekte
  - Festhalten an prozeduraler Entwicklung
    - → includes als Hauptstrukturierungsmittel

-> AOP derzeit ein Randthema

AOP mit PHP 2/3

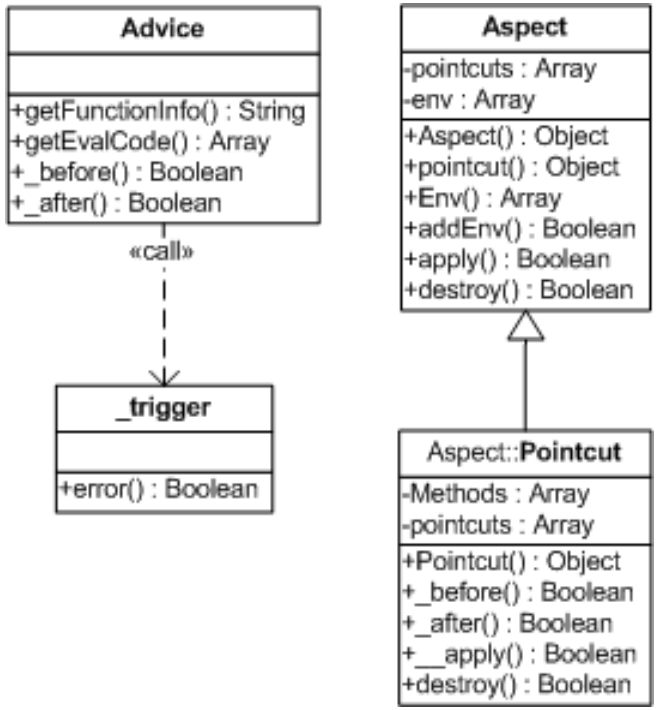
- Vorhandene AOP-Ansätze überwiegend durch Einzelpersonen oder kleine Gruppen unter Open Source entwickelt
  - Prototypen / Machbarkeitsstudien / niedrige Versionsnummern
    - → Bugs / Unvollständigkeiten
    - → Manuelle Konfiguration / geringe Kompatibilität
      - Abhängigkeit von anderen Projekten
    - → Keine Toolunterstützung
    - → Dokumentation oft nicht sehr umfangreich
      - Keine gedruckte Literatur
      - Quellen:
        - » Seiten der Urheber
        - » Opensource-Plattformen
        - » Einschlägige PHP- bzw. Webentwickler-Websites
        - » Blogs

AOP mit PHP 3/3

- Probleme
  - PHP erfordert kein Kompilieren
    - → Übersetzungsvorgang bei statischem Weaving erhöht den Aufwand beim Testen und Ausliefern gegenüber normaler PHP-Entwicklung
  - PHP ist Webtechnologie
    - → Betrieb auf Webservern, die je nach Hoster u.U. nicht oder kaum konfigurierbar sind
      - Dynamisches Weaving mit Hilfe von Spracherweiterungen problematisch
    - → Zugriff durch viele Nutzer
      - Ggf. Performanceprobleme durch Overhead bei dynamischem Weaving

aopinphp 1.2 1/4

- http://www.phpclasses.org/browse/package/2633.html
- AOP-Realisierung mit Hilfe einer Klassenbibliothek



aopinphp 1.2 2/4

- Voraussetzungen
  - PHP 4.1.x oder größer
- Konfiguration
  - aop.lib.php zum Projekt hinzufügen
- Vorgehensweise
  - Erzeugen einer Instanz der Klasse Aspect
  - Festlegen der Pointcuts
  - Definieren der bei before und after durchzuführenden Aktionen
  - Anwenden des Aspekts
  - Festlegen der Stellen, an denen die Adviceaufrufe erfolgen sollen, in den betreffenden Methoden

aopinphp 1.2 3/4

Übersicht

- Was ist PHP?
- Funktionsweise
- Besonderheiten
- AOP mit PHP
- aopinphp
- Bunny Aspects
- Seasar
- aoPHP
- MFAOPHP
- phpaspect
- Fazit

```
<?PHP
    include("aop.lib.php");
    $aspect = new Aspect();
    $pc = $aspect->pointcut("call HelloWorld::HelloWorld or call HelloWorld::sayHello");
    $pc->_before("print 'PreProcess<br />';");
    $pc->_after("print 'PostProcess<br />';");
    Aspect:::apply($aspect);

    Class HelloWorld {
            function HelloWorld() {
                    Advice::_before();
                    echo "HelloWorld created<br/>";
                    Advice::_after();
            }
            function sayHello() {
                    Advice::_before();
                    echo "Hello World<br/>";
                    Advice::_after();
            }
    }

    $myHelloWorld = new HelloWorld();
    $myHelloWorld->sayHello();
?>
```

aopinphp 1.2 4/4

- „manuelles Weaving“ durch den Entwickler:

- Vorteile
  - Kein Übersetzungsvorgang nötig
    - → PHP-typisch direkte Ausführung des geschriebenen Codes
  - Keine Sprach- oder Webservererweiterung
    - → überall ohne Veränderungen an PHP / Webserver lauffähig

- Nachteil
  - Modifikation der Klassen erforderlich, die mit Aspekten versehen werden sollen
    - → geht am Grundgedanken von AOP vorbei

Bunny Aspects Alpha 1/4

- Aspekte auf Objektebene
- http://wiki.jonnay.net/bunny/bunnyaspectsphp

- Konfiguration
  - BunnyAspects.php zum Projekt hinzufügen

- Vorgehensweise
  - Klasse normal implementieren
  - Advice-Methoden anlegen
  - Instanz der Klasse BunnyAspects erzeugen und dabei Objekt der implementierten Klasse im Konstruktor übergeben
    - → Rückgabewert ist ein Objekt mit dem selben Verhalten wie das Ausgangobjekt
  - Mit der weave-Methode des zurückgelieferten Objekts Joinponts definieren

Bunny Aspects Alpha 2/4


```
function logMethodCall($method, $arguments){
    echo "Called $method<br/>";
    echo "Arguments were:<br/>";
    var_dump($arguments);
    return $arguments;
}

function logMethodReturn($method, $returnVal){
    echo "Returning from $method<br/>";
    echo "Return value:<br/>";
    var_dump ($returnVal);
}
```

Bunny Aspects Alpha 3/4


```
class HelloWorld{
    function sayHello($a){
            echo "Hello World<br/>";
            return $a * $a;
    }
}

$myHelloWorld = new HelloWorld();
$myHelloWorld = new BunnyAspect($myHelloWorld);
$myHelloWorld->weave(array('before','sayHello'), 'logMethodCall');
$myHelloWorld->weave(array('after','sayHello'), 'logMethodReturn');
$res = $myHelloWorld->sayHello(5);

Ausgabe:
    Called sayHello
    Arguments were:
    (int) 5
    Hello World
    Returning from sayHello
    Return value:
    (int) 25
```

Bunny Aspects Alpha 4/4

- Manuelles Weaving auf Objekt- statt Klassenebene

- Vorteile
  - Kein Übersetzungsvorgang
    - → Direkte Ausführung des geschriebenen Codes
  - Keine Spracherweiterung
    - → Läuft auf Webservern mit standardmäßiger PHP-Installation / Konfiguration
  - Ohne Anpassung der Implementierung bestehender Klassen verwendbar

- Nachteile
  - Weaving muss im Code veranlasst werden
  - Bei jeder Instanziierung eines Objekts erneutes Weaving erfoderlich
  - Typinformationen des Ausgangsobjekts gehen verloren

Seasar 1/6 1.0.0 beta2

- http://www.seasar.org/
- Definieren von Aspekten mit XML-Dateien und Interceptoren
- Voraussetzungen
  - PHP 5.x
- Konfiguration
  - s2container.php5-Dateien zum Projekt hinzufügen

Seasar 2/6 1.0.0 beta2

- Vorgehensweise
  - Klasse wie gewohnt implementieren
  - Aspekt in einer XML-Datei (Dicon) definieren und Interceptor anlegen
  - Mit Hilfe der XML-Datei einen S2Container erzeugen
  - Instanz der Klasse über die getComponent-Methode des Containers holen

```
<?php
   class HelloWorld {
                  function sayHello(){
                   echo "Hello World<br/>";
            }
   }
?>
```

Seasar 3/6 1.0.0 beta2


```
<components>
    <component name="traceInterceptor" class="TraceInterceptor"/> <component
    class="HelloWorld">
            <aspect pointcut="sayHello"> traceInterceptor </aspect> </component>
</components>
```

```
<?php
    require_once('s2container.inc.php');
    $container = S2ContainerFactory::create("Trace.dicon“);
    $myHelloWorld = $container->getComponent('HelloWorld');
    $myHelloWorld->sayHello();
?>
```

Ausgabe:

```
    BEGIN HelloWorld#sayHello() Hello World
    END HelloWorld#sayHello() :
```

Seasar 4/6 1.0.0 beta2

- Besonderheit
  - Definition von Join Points und Advices erfolgt über sogenannte Interceptoren
    - → TraceInterceptor: Loggen von Methodenaufrufen
    - → ThrowsInterceptor: Ausnahmebehandlung
    - → DelegateInterceptor: Umleiten von Methodenaufrufen
    - → (…)
  - Spezielles Verhalten wird durch Ableiten und Erweitern der vorhandenen Interceptor-Klassen realisiert

```
<?php
    class HandleThrowableInterceptor extends ThrowsInterceptor {
            public function handleThrowable(Exception $t,
                    MethodInvocation $invocation){
                    throw new S2RuntimeException("ESSR0007",
                    array("arg")); }
} ?>
```

  - Eigene Interceptoren können durch Ableiten von AbstractInterceptor realisiert werden

Seasar 4/6 1.0.0 beta2

- Alternative Verwendung ohne XML: Weaving durch AOP-Proxy

```
<?php
    $pointcut = new PointcutImpl(array("sayHello"));
    $aspect = new AspectImpl(new TraceInterceptor(), $pointcut);
    $aopProxy = new AopProxy('HelloWorld', array($aspect));
    $myHelloWorld = $aopProxy->create();
    $myHelloWorld->sayHello();
?>
```

Seasar 6/6 1.0.0 beta2

- Aspektrealisierung über Interceptoren

- Vorteile
  - Kein Übersetzungvorgang erforderlich
    - → Direkte Ausführung des geschriebenen Codes
  - Keine Spracherweiterung
    - → Läuft auf Webservern mit standardmäßiger PHP-Installation / Konfiguration
  - Überdurchschnittlich mächtig
  - Ohne Anpassung der Implementierung bestehender Klassen verwendbar

- Nachteile
  - Weaving muss im Code veranlasst werden
  - Unterscheidet sich bei der Anwendung erheblich von den anderen AOP-Implementierungen

aoPHP 1/7

- http://www.aophp.net
- Java Precompiler
- AOP durch Veränderung der Klassen mittels einer Java-Erweiterung
- Voraussetzungen
  - PHP 4.3.0 oder höher
  - Java 1.5 oder höher
- Konfiguration
  - aoPHP Binaries installieren
  - Verzeichnispfade anpassen
  - aophpexec.oaphp und .htaccess in Webfolder kopieren

aoPHP 2/7

- Vorgehensweise
  - Normales Implementieren der PHP-Klassen
  - Definieren von Aspekten in aophp-Files
    - → before
    - → after
    - → around
  - Zuweisung der Aspekte durch Angabe von Funktionsname und Parametern
  - Klassen und Aspekte werden als neue Dateien in einem definierten Verzeichnis zusammengewebt

aoPHP 3/7


```
<?aophp filename="aotest.aophp,aotest2.aophp" debug="off"

    function sub($x,$y){
        return $x-$y;
    }

    function div($a,$b){
        return $a/$b;
    }

    function say($text){
        echo "PHP Says: $text <br>";
    }

    say("Hello World");
    echo "<br><br>";

    $n1 = 5;
    $n2 = 10;
    echo "N1: $n1 | N2: $n2<br>";

     $diff = sub($n1,$n2);
    echo "Difference of N1 & N2: $diff <br>";

    $quotient = div($n1,0);
    echo "Quotient of N1 & 0: $quotient <br>";
?>
```

aoPHP 4/7

Übersicht

- Was ist PHP?
- Funktionsweise
- Besonderheiten
- AOP mit PHP
- aopinphp
- Bunny Aspects
- Seasar
- aoPHP
- MFAOPHP
- phpaspect
- Fazit

```
around(): execr(div($a,$b)){
    if($b == 0){
        echo "<font color=red>Cant Divide by 0, Returning -1</font><br>";
        return -1;
    } else {
        echo "<font color=red>Division Allowed, Proceeding</font><br>";
        proceed($a,$b);
    }
}

before(): exec(say($text)){
    echo "<font color=red>Before Talking</font></br>";
}

after(): exec(say($text)){
    echo "<font color=red>After Talking</font></br>";
}

around(): exec(say($text)){
    echo "<font color=red>AOPHP Says: PHP Tried To Say '$text', But I Said No.</font></br>";
}
after(): execr(add($x,$y)) | execr(div($a,$b)){
    echo "<font color=red>I Just did Some Math</font></br>";
}
```

aoPHP 5/7

Before Talking
AOPHP Says: PHP Tried To Say 'Hello World', But I Said No.
After Talking

N1: 5 | N2: 10
Im About To Add/Sub 5 & 10
Im Going to Increment 5 by 1, then Add it
I Just did Some Math
Total of N1 & N2: 16

Cant Divide by 0, Returning -1
I Just did Some Math
Quotient of N1 & 0: -1

aoPHP 6/7

- Vorteile
  - Ohne Anpassung der Implementierung bestehender Klassen verwendbar
  - Weaving ohne Aktivierung durch den Entwickler
- Nachteile
  - Betriebsystemabhängig, nur Linux OS
  - Root Level System Zugriff nötig
  - Läuft nur auf Apache Web Server
  - Benötigt Java JDK ab 1.5

aoPHP 7/7

aspectPHP

http://www.cs.toronto.edu/~yijun/aspectPHP/

- aoPHP weiterentwickelt in C
- Erweiterung für den Apache-Server
- Plattformunabhängig
- Gleiche Funktionalität

MFAOPHP 0.9 1/6

- http://www.mfaop.com/
- Modifizieren von PHP-Klassen zur Laufzeit
- Voraussetzungen
  - PHP 5.0.3 oder höher
- Konfiguration
  - PECL-Paket (http://pecl.php.net/) Classkit 0.4 installieren und in php.ini eintragen

classkit support	enabled
version	0.4

  - Aspect.php, JoinPoint.php und PointCut.php zum Projekt hinzufügen

MFAOPHP 0.9 2/6

- Vorgehensweise
  - Implementieren der mit Aspekten zu versehenden Klassen wie gewöhnlich
  - Erzeugen von Pointcuts
  - Hinzufügen von Joinpoints zu den Pointcuts
  - Aspekte mit den Pointcuts und zugehöriger Logik als Übergabeparameter anlegen

MFAOPHP 0.9 3/6

Übersicht

- Was ist PHP?
- Funktionsweise
- Besonderheiten
- AOP mit PHP
- aopinphp
- Bunny Aspects
- Seasar
- aoPHP
- MFAOPHP
- phpaspect
- Fazit

```
<?PHP
    require_once('./JoinPoint.php');
    require_once('./PointCut.php');
    require_once('./Aspect.php');

    class HelloWorld {
            function sayHello() {
                    echo "Hello World<br/>";
            }
            function anotherHello() {
                    echo "Hello World<br/>";
                return "Hello World<br/>";
             }
    }
```

MFAOPHP 0.9 4/6


```
$myHelloWorld = new HelloWorld();

$testPointCut = new PointCut();
$testPointCut->addJoinPoint('HelloWorld', 'sayHello');
$testPointCut->addJoinPoint('HelloWorld', 'anotherHello');
//alternativ: $testPointCut->addJoinPoint('HelloWorld', AllMethods);
$test1 = new Aspect ($testPointCut, before, 'echo "Before
        $MethodName<br/>";');
$test2 = new Aspect ($testPointCut, after, 'echo "After  $MethodName<br/>";');
$test3 = new Aspect ($testPointCut, around, '$Return = "New return value of
        $MethodName<br/>";');

$myHelloWorld->sayHello();
echo $myHelloWorld->anotherHello();
?>
```

MFAOPHP 0.9 5/6

- Classkit
  - Hinzufügen von Methoden zu Klassen: classkit_method_add
  - Methoden von einer Klasse zur einer anderen Kopieren: classkit_method_copy
  - Verändern des Codes von Methoden: classkit_method_redefine
  - Methoden entfernen: classkit_method_remove
  - Methoden umbenennen: classkit_method_rename

```
<?php
    class Foo {
        function sayFoo() {
                echo "foo!\n";
        }
    }

    $myFoo = new Foo();
    classkit_method_add( ,Foo', 'add', '$num1 , $num2', 'return $num1 + $num2;',
        CLASSKIT_ACC_PUBLIC );
    echo $myFoo->add(12, 4);
?>
```

MFAOPHP 0.9 6/6

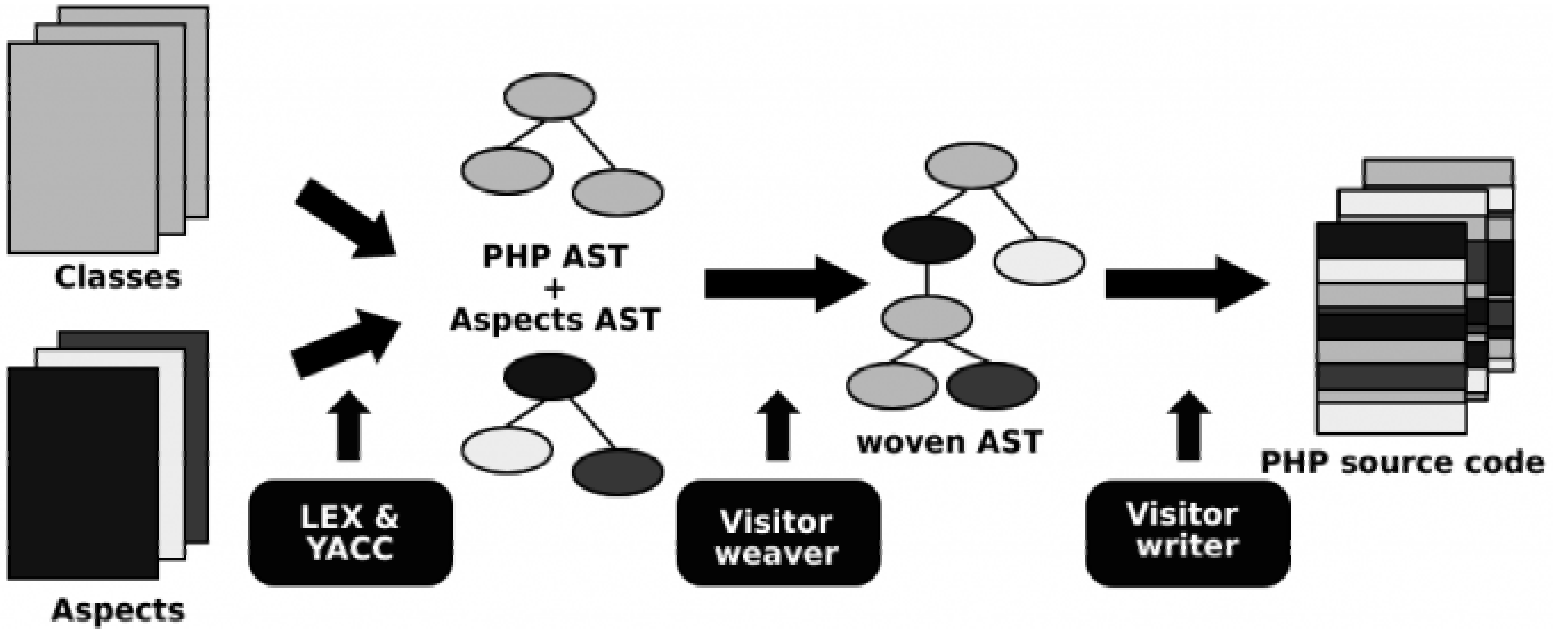
- Weaving durch Modifizieren von Klassen zur Laufzeit

- Vorteile
  - Kein Übersetzungsvorgang nötig
    - → Direkte Ausführung des geschriebenen Codes
  - Relativ einfach einsetzbar, da lediglich Classkit installiert werden muss
  - Ohne Anpassung der Implementierung bestehender Klassen verwendbar

- Nachteile
  - Ohne Zugriff auf PHP-Installation / Konfiguration nicht einsetzbar
  - Entwicklung von Classkit inzwischen eingestellt
    - → Portierung von MFAOPHP auf Nachfolgeprodukt Runkit steht noch aus

phpaspect 0.3 1/9

- http://www.phpaspect.org/
- PHP-Spracherweiterung
- Weaving mit Hilfe eines in PHP geschriebenen Compilers



LEX / YACC: Compilerbauwerkzeuge AST: Abstract Syntax Tree

phpaspect 0.3 2/9

- Voraussetzungen
  - PHP 5.x
- Konfiguration
  - Für Compiler
    - → Pear-Pakete installieren (http://www.pear.php.net/)
      - Console_Getopt (aktuell: Version 1.2)
      - Console_Progressbar (aktuell: Version 0.4.0beta)
      - PHP_Beautifier (aktuell: Version 0.1.8)
      - Console_Color (nur für phpaspect 0.2, aktuell: Version 0.0.3)
  - Zur Ausführung des erzeugten Codes
    - → phpaspect_lib ins Pear-Verzeichnis kopieren
      oder
    - → Pfad zu phpaspect_lib in die php.ini eintragen
      oder
    - → set_include_path verwenden

phpaspect 0.3 3/9

- Vorgehensweise
  - Implementieren der mit Aspekten zu versehenden Klassen wie gewöhnlich
  - Definieren der Pointcuts mit der
    - → before
    - → after
    - → und around

    auszuführenden Logik in einem Programmodul mit dem Schlüsselwort aspect
  - Weaving mit phpaspect-Compiler
    - → Aufruf: php Pfad\phpaspect.php Quellpfad Zielpfad

phpaspect 0.3 4/9

Übersicht

- Was ist PHP?
- Funktionsweise
- Besonderheiten
- AOP mit PHP
- aopinphp
- Bunny Aspects
- Seasar
- aoPHP
- MFAOPHP
- phpaspect
- Fazit

```
<?php
    class HelloWorld{
            public function sayHello(){
                    echo "Hello World<br/>";
            }
    }

    $myHelloWorld = new HelloWorld();
    $myHelloWorld->sayHello();
?>
```

```
<?php
    aspect TraceHello{
            pointcut logSayHello:exec(public HelloWorld->sayHello(0));
            after logSayHello{
                    echo "sayHello executed<br/>";
            }
    }
?>
```

phpaspect 0.3 5/9


```
(…)
class  HelloWorld {
        public  function  sayHello() {
                echo  "Hello World<br/>";
                $thisJoinPoint =  new
                        ExecJoinPoint(array('name'=>'logSayHello',
                        'class'=>'HelloWorld',  'method'=>'sayHello',
                        'arguments'=>'0'),  '2',  'HelloWorld.php',  array(),
                        $this);
                echo  "sayHello executed<br/>";
                unset($thisJoinPoint);
        }
}
(…)
```

phpaspect 0.3 6/9

- Möglichkeiten
  - Join Points
    - → Methodenaufruf
      - Statisch: pointcut logTotalAmount:call(Catalog::getPrice($ref));
      - Dynamisch: pointcut logTotalAmount:call(Order->addItem(2));
    - → Methodenausführung: pointcut logAddItem:exec(public Order::addItem(2));
    - → Attributzugriff (statisch & dynamisch)
      - Lesend: pointcut TraceSet:set(Order->$catalog);
      - Schreibend: pointcut TraceSet:get(Catalog::$priceList);
    - → Konstruktoraufruf: pointcut TraceConstruct:new(Order(1));
    - → Zerstörung eines Objekts: pointcut UpdateStates:unset(Order);
    - → Ausnahmebehandlung: poincut catchOrderException:catch(OrderException);
  - Wildcards: pointcut logAddItem:exec(* Order::addItem(*));
  - Kombinieren von Join Points

```
before JoinPointA || JoinPointB || JoinPointC {
        echo "Before A, B or C joinpoint"; }
```

  - Inter-Type Declarations

```
public function Foo*::add($num1, $num2){
        return $num1 + $num2;
}
```

phpaspect 0.3 7/9

- Toolbasiertes Weaving vor der Codeausführung:

- Vorteile
  - Auf Webservern relativ einfach einsetzbar, da lediglich die phpaspect_lib eingebunden werden muss (bei Verwendung von set_include_path ohne Konfigurationsänderung möglich)
  - Ohne Anpassung der Implementierung bestehender Klassen verwendbar
  - Überdurchschnittlich mächtig

- Nachteile
  - Übersetzungsvorgang erforderlich
    - → Widerspricht der bei PHP üblichen Vorgehensweise
    - → Zusätzlicher Arbeitsschritt erschwert das Testen und Publizieren von PHP-Seiten / Anwendungen
  - Sehr frühes Enwicklungsstadium
    - → Bugs
    - → Kryptische Fehlermeldungen

phpaspect 0.3 8/9

- phpaspect 0.1
  - Definition von Aspekten mit XML
  - Weaving bei der Codeausführung mit der Klasse Aspect

```
<aspect name="Trace">
    <pointcut name="HelloWorld" type="call" select="$myHelloWorld->sayHello">
            <before>echo "HelloWorld::sayHello() will be executed";</before>
            <after>echo "HelloWorld::sayHello() has been executed";</after>
    </pointcut>
</aspect>

<?php
    require "../../libs/aspect.class.php";
    $aspect = new Aspect(__FILE__, realpath('aspects/helloworld.xml'));
    $aspect->setClass('HelloWorld', 'classes/helloworld.class.php');
    $aspect->weave();
    $myHelloWorld = new HelloWorld;
    $myHelloWorld->sayHelloWorld();
?>
```

phpaspect 0.3 9/9

- phpaspect 0.1
  - Mit Aspekten auszustattende Klasse bleibt unverändert
  - Die Ausführung des aufrufenden Skripts wird abgebrochen und stattdessen wird ein unter Berücksichtigung des Ausgangscodes und der Aspekte generiertes Skript abgearbeitet:

```
<?php
    class HelloWorld {
        function sayHello() {
                echo "Hello World";
        }
    }
?>

<?php
    require "../../libs/aspect.class.php";
    $hello=new HelloWorld;
    echo "HelloWorld::sayHello() will be executed";
    $hello->sayHello();
    echo "HelloWorld::sayHello() has been executed";
?>
```

Fazit

- AOP für den produktiven Einsatz derzeit problematisch
  - AOP-Lösungen überwiegend noch unausgereift
  - Einsatz oft unkomfortabel
  - Fehlende Toolunterstützung
  - In einigen Fällen erweiterte Rechte auf dem PHP- bzw. Webserver erforderlich
  - unbestimmte Lebensdauer der AOP-Projekte
- Ausblick
  - Runkit scheint zu einer zentralen PHP-Erweiterung zu entwickeln
    - → Wahrscheinlich breite Installationsbasis auch bei Hostern
    - → Erlaubt das Verändern von Klassen zur Laufzeit
    - → Daher ideale Grundlage für dynamisches Weaving
    - → Projekte: MFAOPHP (Portierung von Classkit auf Runkit geplant)
      AspectPHP (http://www.sebastian-bergmann.de/AspectPHP/)

Fragen

Übersicht

- Was ist PHP?
- Funktionsweise
- Besonderheiten
- AOP mit PHP
- aopinphp
- Bunny Aspects
- Seasar
- aoPHP
- MFAOPHP
- phpaspect
- Fazit

Referenzen 1/3

- aoPHP
  - http://www.aophp.net/
  - http://sourceforge.net/projects/aophp/
- aopinphp
  - http://www.phpclasses.org/browse/package/2633.html
  - http://pear.php.net/pepr/pepr-proposal-show.php?id=315
  - http://www.weberdev.com/ViewArticle/442
  - http://www.weberdev.com/get_example-4237.html
  - http://enterprise.phpmagazine.net/2005/12/aspect_oriented_programming_fo.html
  - http://freshmeat.net/projects/aophp/?branch_id=61075&release_id=219924
- aspectPHP
  - http://www.cs.toronto.edu/~yijun/aspectPHP/
- Bunny Aspects
  - http://blog.jonnay.net/archives/637-Aspect-Oriented-Programming-in-PHP-as-a-contrast-to-other-languages..html
  - http://wiki.jonnay.net/bunny/bunnyaspectsphp

Referenzen 2/3

- phpaspect
  - http://phpaspect.org/wiki/doku.php?id=phpaspect
  - http://groups.google.com/group/phpaspect
- MFAOPHP
  - http://www.mfaop.com/
- Seasar
  - http://www.seasar.org/en/php5/index.html
- Sonstiges zu AOP in PHP
  - http://jaxn.org/blog/category/technology/php/aspect_oriented/
  - http://osteele.com/archives/2004/08/web-aop
  - http://www.sebastian-bergmann.de/blog/archives/573-Current-State-of-AOP-for-PHP.html
  - http://www.sebastian-bergmann.de/AspectPHP/
  - http://www.phpcompiler.org/spinoffs/index.html

Referenzen 3/3

- PHP
  - http://de.wikipedia.org/wiki/Php
  - http://www.ister.org/code/article/de/phpbench51.xhtml
  - http://www.php.net/manual/de/
  - http://www.php.net/usage.php
  - http://www.php.net/manual/de/language.oop.php
  - http://www.php.net/manual/de/language.oop5.php
  - http://www.apachefriends.org/de/index.html