

Masterseminar Informatik

Aspektorientierte Programmierung mit



hochschule mannheim

Ralf Haingärtner / Markus Pech

04.05.2006



Agenda

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

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



Was ist PHP?

„**PHP: Hypertext Preprocessor**“

früher

Funktionsweise

„**Personal Home Page Tools**“

Besonderheiten

AOP mit PHP

- PHP ist eine Skriptsprache
 - Weniger mächtig als “echte” Programmiersprachen
 - Vorteilhaft bei der schnellen Erstellung von kleinen Programmen
 - Nachteilig bei größeren Projekten

aopinphp

Bunny Aspects

Seasar

- PHP ist Webtechnologie
 - Dient hauptsächlich der Erstellung dynamischer Websites...
 - oder ganzer Webanwendungen
 - Auch für Offline-Skripte benutzbar

aoPHP

MFAOPHP

phpaspect

- PHP wird serverseitig interpretiert
 - Keine Anforderungen an den Client (Browser)
 - Kein Quellcode einsehbar
 - Ausgabe von HTML, PDF, Bildern etc.

Fazit



Was ist PHP ? 2/4

Was ist PHP?

Funktionsweise

Besonderheiten

AOP mit PHP

aopinphp

Bunny Aspects

Seasar

aoPHP

MFAOPHP

phpaspect

Fazit

- 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

- ▶ Was ist PHP?
 - PHP 1.0 erschienen Juni 1995
 - Sammlung von Perl-Skripten
- ▶ Funktionsweise
 - PHP 2.0.0 bzw. PHP/FI, erschienen November 1997
- ▶ Besonderheiten
 - PHP 3.0.0, erschienen Juni 1998
- ▶ AOP mit PHP
 - PHP 4.0.0, erschienen Mai 2000
 - Einfache Objektorientierte Programmierung
- ▶ aopinphp
 - Zend Engine 1
- ▶ Bunny Aspects
 - PHP 5.0.0, erschienen Juli 2004
 - Objektorientiertes Programmieren
- ▶ Seasar
 - Zend Engine 2
- ▶ aoPHP
 - PHP 5.1.0, erschienen November 2005
- ▶ MFAOPHP
 - PHP 6.0.0 angekündigt für das dritte Quartal 2006
 - umfassende Unicode-Unterstützung
- ▶ phpaspect
- ▶ Fazit

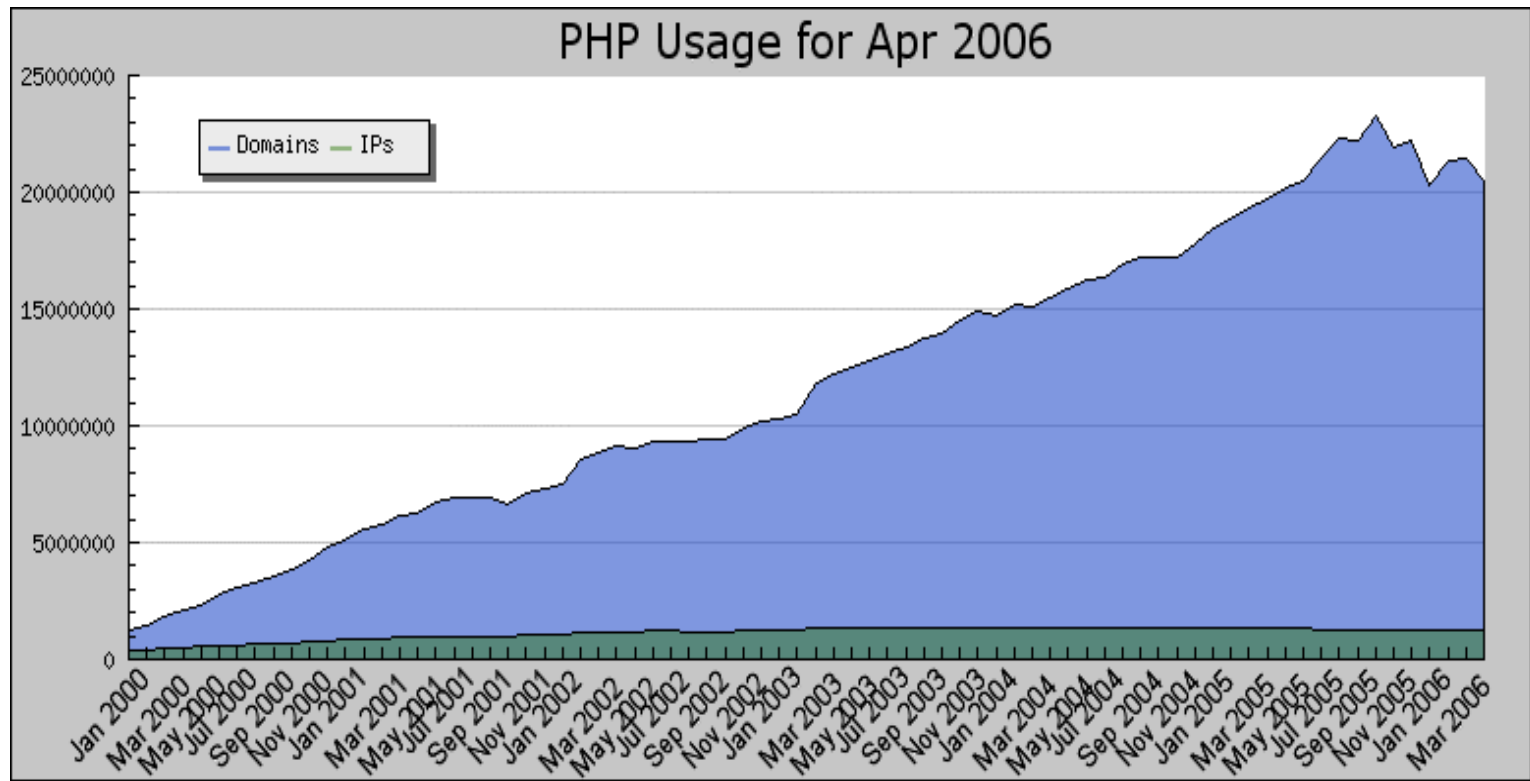


Was ist PHP ? 4/4

Verbreitung von PHP

Was ist PHP?

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



Folie 5



Funktionsweise von PHP 1/3

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



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:

▶ Was ist PHP?

▶ Funktionsweise

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

▶ Besonderheiten

Code eingebettet in HTML:

▶ AOP mit PHP

▶ aopinphp

```
<html>
```

▶ Bunny Aspects

```
<head><title> Hello World!-Beispiel</title></head>
```

▶ Seasar

```
<body>
```

▶ aoPHP

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

▶ MFAOPHP

```
</body>
```

▶ phpaspect

```
</html>
```

▶ Fazit



▶ Was ist PHP?

▶ Funktionsweise

▶ Besonderheiten

▶ AOP mit PHP

▶ aopinphp

▶ Bunny Aspects

▶ Seasar

▶ aoPHP

▶ MFAOPHP

▶ phpaspect

▶ Fazit

- **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

- ▶ Was ist PHP?
- ▶ Funktionsweise
- ▶ **Besonderheiten**
 - Keine Persistenz
 - Skripte arbeiten nur bis zur Fertigstellung der Ausgabe
 - Variablen und Objekte verlieren mit dem Ende der Ausführung ihres Skripts die Gültigkeit
- ▶ AOP mit PHP
 - Sicherung von Variablen und Objekten über Serialisierung in einem Session-Array
- ▶ aopinphp
- ▶ Bunny Aspects
 - PHP-Applikationen bestehen aus Einzelskripten, Kommunikation per gegenseitigem Aufruf oder Einbinden
- ▶ Seasar
- ▶ aoPHP
- ▶ MFAOPHP
 - Übermittlung von Parametern über GET- bzw. POST-Array
- ▶ phpaspect
- ▶ Fazit



Besonderheiten von PHP 2/6

- ▶ Was ist PHP?
- ▶ Funktionsweise
- ▶ **Besonderheiten**
 - Meist Fremdhosting
 - kein Zugriff auf Serverkonfiguration
 - Einbinden von Erweiterungen schwierig
- ▶ AOP mit PHP
 - LAMP/WAMP-Pakete
 - Linux / Windows
 - Apache Webserver
 - MySql
 - PHP
- ▶ aopinphp
- ▶ Bunny Aspects
- ▶ Seasar
- ▶ aoPHP
 - Prozedurale Programmierung
 - Einfache Skripte
 - Persönliche Webseiten
 - Kleine Projekte
- ▶ MFAOPHP
- ▶ phpaspect
- ▶ Fazit



Besonderheiten von PHP 3/6

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

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

▶ Was ist PHP?

▶ Funktionsweise

▶ Besonderheiten

▶ AOP mit PHP

▶ aopinphp

▶ Bunny Aspects

▶ Seasar

▶ aoPHP

▶ MFAOPHP

▶ phaspect

▶ Fazit

```
<?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);
    }
}
?>
```



Besonderheiten von PHP 5/6

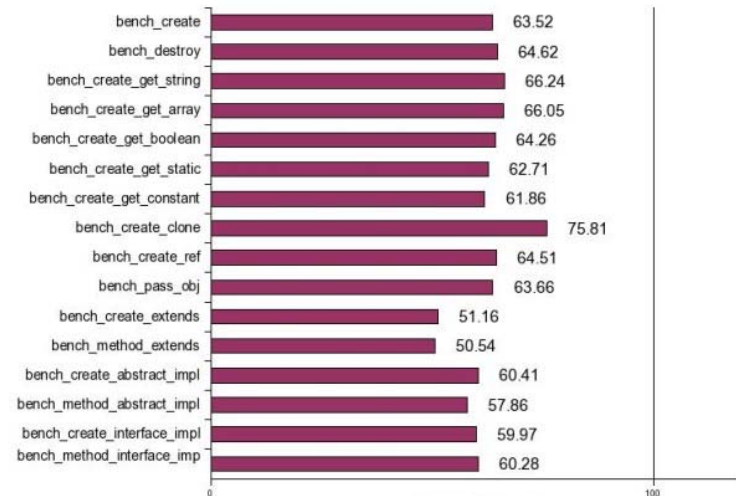
PHP 5.1

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.

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



PHP 5.0.0



PHP 5.1.0



Fragen

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





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

- 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



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

- 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



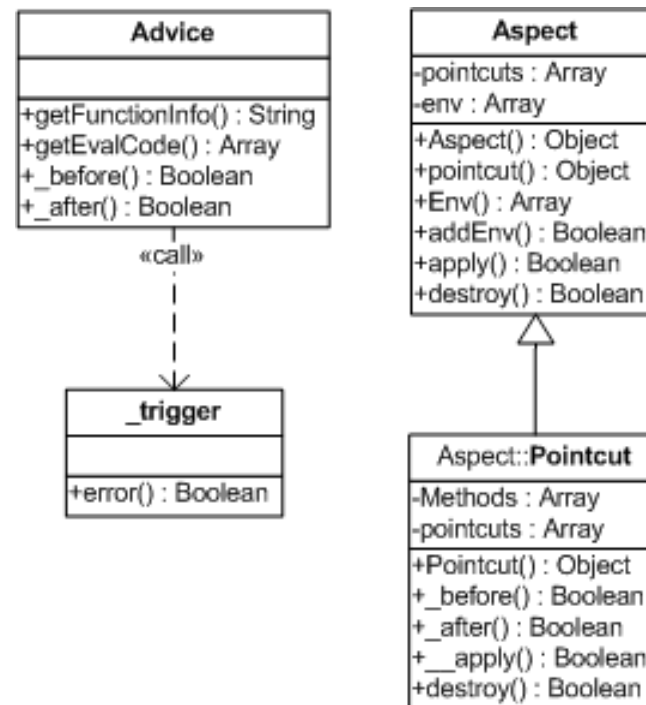
- ▶ Was ist PHP?
- ▶ Funktionsweise
- ▶ Besonderheiten
- ▶ AOP mit PHP
- ▶ aopinphp
- ▶ Bunny Aspects
- ▶ Seasar
- ▶ aoPHP
- ▶ MFAOPHP
- ▶ phaspect
- ▶ Fazit

- 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



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

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





- ▶ Was ist PHP?
 - Voraussetzungen
 - PHP 4.1.x oder größer
- ▶ Funktionsweise
- ▶ Besonderheiten
 - Konfiguration
 - aop.lib.php zum Projekt hinzufügen
- ▶ AOP mit PHP
- ▶ aopinphp
- ▶ Bunny Aspects
 - 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
- ▶ Seasar
- ▶ aoPHP
- ▶ MFAOPHP
- ▶ phaspect
- ▶ Fazit



aopinphp 1.2 3/4

▶ Was ist PHP?	<?PHP
▶ Funktionsweise	include("aop.lib.php");
▶ Besonderheiten	\$aspect = new Aspect();
▶ AOP mit PHP	\$pc = \$aspect->pointcut("call HelloWorld::HelloWorld or call HelloWorld::sayHello");
▶ aopinphp	\$pc->_before("print 'PreProcess ';");
▶ Bunny Aspects	\$pc->_after("print 'PostProcess ';");
▶ Seasar	Aspect:::apply(\$aspect);
▶ aoPHP	Class HelloWorld {
▶ MFAOPHP	function HelloWorld() {
▶ phpaspect	Advice::_before();
▶ Fazit	echo "HelloWorld created ";
	Advice::_after();
	}
	function sayHello() {
	Advice::_before();
	echo "Hello World ";
	Advice::_after();
	}
	}
	\$myHelloWorld = new HelloWorld();
	\$myHelloWorld->sayHello();
	?>



▶ Was ist PHP?

▶ Funktionsweise

▶ Besonderheiten

▶ AOP mit PHP

▶ aopinphp

▶ Bunny Aspects

▶ Seasar

▶ aoPHP

▶ MFAOPHP

▶ phaspect

▶ Fazit

- „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

- ▶ Was ist PHP?
 - ▶ Funktionsweise
 - ▶ Besonderheiten
 - ▶ AOP mit PHP
 - ▶ aopinphp
 - ▶ **Bunny Aspects**
 - ▶ Seasar
 - ▶ aoPHP
 - ▶ MFAOPHP
 - ▶ phaspect
 - ▶ Fazit
- 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 Ausgangsobjekt
 - Mit der weave-Methode des zurückgelieferten Objekts Joinpoints definieren



Bunny Aspects Alpha 2/4

▶ Was ist PHP?	function logMethodCall(\$method, \$arguments){
▶ Funktionsweise	echo "Called \$method ";
▶ Besonderheiten	echo "Arguments were: ";
▶ AOP mit PHP	var_dump(\$arguments);
▶ aopinphp	return \$arguments;
▶ Bunny Aspects	}
▶ Seasar	function logMethodReturn(\$method, \$returnVal){
▶ aoPHP	echo "Returning from \$method ";
▶ MFAOPHP	echo "Return value: ";
▶ phpaspect	var_dump (\$returnVal);
▶ Fazit	}



Bunny Aspects Alpha 3/4

▶ Was ist PHP?	class HelloWorld{
▶ Funktionsweise	function sayHello(\$a){
▶ Besonderheiten	echo "Hello World ";
▶ AOP mit PHP	return \$a * \$a;
▶ aopinphp	}
▶ Bunny Aspects	} \$myHelloWorld = new HelloWorld();
▶ Seasar	\$myHelloWorld = new BunnyAspect(\$myHelloWorld);
▶ aoPHP	\$myHelloWorld->weave(array('before','sayHello'), 'logMethodCall');
▶ MFAOPHP	\$myHelloWorld->weave(array('after','sayHello'), 'logMethodReturn');
▶ phpaspect	\$res = \$myHelloWorld->sayHello(5);
▶ Fazit	Ausgabe:
	Called sayHello
	Arguments were:
	(int) 5
	Hello World
	Returning from sayHello
	Return value:
	(int) 25

Folie 25



Bunny Aspects Alpha 4/4

▶ Was ist PHP?

▶ Funktionsweise

▶ Besonderheiten

▶ AOP mit PHP

▶ aopinphp

▶ Bunny Aspects

▶ Seasar

▶ aoPHP

▶ MFAOPHP

▶ phaspect

▶ Fazit

- 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 erforderlich
- Typinformationen des Ausgangsobjekts gehen verloren



Seasar 1/6 1.0.0 beta2

- ▶ Was ist PHP?
 - ▶ Funktionsweise
 - ▶ Besonderheiten
 - ▶ AOP mit PHP
 - ▶ aopinphp
 - ▶ Bunny Aspects
 - ▶ Seasar
 - ▶ aoPHP
 - ▶ MFAOPHP
 - ▶ phaspect
 - ▶ Fazit
- <http://www.seasar.org/>
 - Definieren von Aspekten mit XML-Dateien und Interceptoren
 - Voraussetzungen
 - PHP 5.x
 - Konfiguration
 - s2container.php5-Dateien zum Projekt hinzufügen



- ▶ Was ist PHP?
 - ▶ Funktionsweise
 - ▶ Besonderheiten
 - ▶ AOP mit PHP
 - ▶ aopinphp
 - ▶ Bunny Aspects
 - ▶ Seasar
 - ▶ aoPHP
 - ▶ MFAOPHP
 - ▶ phaspect
 - ▶ Fazit
- 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
";
 }
}
?>
```



# Seasar 3/6 1.0.0 beta2

|                  |                                                                                                                                                    |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
|                  | <code>&lt;components&gt;</code>                                                                                                                    |
| ▶ Was ist PHP?   | <code>&lt;component name="traceInterceptor" class="TraceInterceptor"/&gt; &lt;component class="HelloWorld"&gt;</code>                              |
| ▶ Funktionsweise | <code>&lt;aspect pointcut="sayHello"&gt; traceInterceptor &lt;/aspect&gt; &lt;/component&gt;</code>                                                |
| ▶ Besonderheiten | <code>&lt;/components&gt;</code>                                                                                                                   |
| ▶ AOP mit PHP    |                                                                                                                                                    |
| ▶ aopinphp       | <code>&lt;?php</code>                                                                                                                              |
| ▶ Bunny Aspects  | <code>require_once('s2container.inc.php');</code>                                                                                                  |
| ▶ Seasar         | <code>\$container = S2ContainerFactory::create("Trace.dicon");</code><br><code>\$myHelloWorld = \$container-&gt;getComponent('HelloWorld');</code> |
| ▶ aoPHP          | <code>\$myHelloWorld-&gt;sayHello();</code>                                                                                                        |
| ▶ MFAOPHP        | <code>?&gt;</code>                                                                                                                                 |
| ▶ phpaspect      |                                                                                                                                                    |
| ▶ Fazit          | Ausgabe:<br><code>BEGIN HelloWorld#sayHello() Hello World</code><br><code>END HelloWorld#sayHello() :</code>                                       |



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

- 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



- Alternative Verwendung ohne XML: Weaving durch AOP-Proxy

▶ Was ist PHP?

▶ Funktionsweise

▶ Besonderheiten

▶ AOP mit PHP

▶ aopinphp

▶ Bunny Aspects

▶ Seasar

▶ aoPHP

▶ MFAOPHP

▶ phpaspect

▶ Fazit

<?php

```
$pointcut = new PointcutImpl(array("sayHello"));
```

```
$aspect = new AspectImpl(new TraceInterceptor(), $pointcut);
```

```
$aopProxy = new AopProxy('HelloWorld', array($aspect));
```

```
$myHelloWorld = $aopProxy->create();
```

```
$myHelloWorld->sayHello();
```

?>



▶ Was ist PHP?

▶ Funktionsweise

▶ Besonderheiten

▶ AOP mit PHP

▶ aopinphp

▶ Bunny Aspects

▶ Seasar

▶ aoPHP

▶ MFAOPHP

▶ phpaspect

▶ Fazit

- Aspektrealisierung über Interceptoren

- Vorteile

- Kein Übersetzungsvorgang 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



▶ Was ist PHP?

▶ Funktionsweise

▶ Besonderheiten

▶ AOP mit PHP

▶ aopinphp

▶ Bunny Aspects

▶ Seasar

▶ **aoPHP**

▶ MFAOPHP

▶ phaspect

▶ Fazit

- <http://www.aopphp.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.oapphp und .htaccess in Webfolder kopieren



- ▶ Was ist PHP?
- ▶ Funktionsweise
- ▶ Besonderheiten
- ▶ AOP mit PHP
- ▶ aopinphp
  - before
  - after
  - around
- ▶ Bunny Aspects
  - Zuweisung der Aspekte durch Angabe von Funktionsname und Parametern
- ▶ Seasar
  - Klassen und Aspekte werden als neue Dateien in einem definierten Verzeichnis zusammengewebt
- ▶ aoPHP
- ▶ MFAOPHP
- ▶ phaspect
- ▶ Fazit



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

|                  |                                                                          |
|------------------|--------------------------------------------------------------------------|
| ▶ Was ist PHP?   | function sub(\$x,\$y){<br>return \$x-\$y;<br>}                           |
| ▶ Funktionsweise |                                                                          |
| ▶ Besonderheiten | function div(\$a,\$b){<br>return \$a/\$b;<br>}                           |
| ▶ AOP mit PHP    |                                                                          |
| ▶ aopinphp       | function say(\$text){<br>echo "PHP Says: \$text <br>";<br>}              |
| ▶ Bunny Aspects  | say("Hello World");<br>echo "<br><br>";                                  |
| ▶ Seasar         |                                                                          |
| ▶ aoPHP          | \$n1 = 5;<br>\$n2 = 10;<br>echo "N1: \$n1   N2: \$n2<br>";               |
| ▶ MFAOPHP        |                                                                          |
| ▶ phpaspect      | \$diff = sub(\$n1,\$n2);<br>echo "Difference of N1 & N2: \$diff <br>";   |
| ▶ Fazit          | \$quotient = div(\$n1,0);<br>echo "Quotient of N1 & 0: \$quotient <br>"; |

?>



▶ Was ist PHP?	<pre>around(): execr(div(\$a,\$b)){     if(\$b == 0){         echo "&lt;font color=red&gt;Cant Divide by 0, Returning -1&lt;/font&gt;&lt;br&gt;";         return -1;     } else {         echo "&lt;font color=red&gt;Division Allowed, Proceeding&lt;/font&gt;&lt;br&gt;";         proceed(\$a,\$b);     } }</pre>
▶ Funktionsweise	
▶ Besonderheiten	
▶ AOP mit PHP	
▶ aopinphp	<pre>before(): exec(say(\$text)){     echo "&lt;font color=red&gt;Before Talking&lt;/font&gt;&lt;/br&gt;"; }</pre>
▶ Bunny Aspects	
▶ Seasar	<pre>after(): exec(say(\$text)){     echo "&lt;font color=red&gt;After Talking&lt;/font&gt;&lt;/br&gt;"; }</pre>
▶ aoPHP	<pre>around(): exec(say(\$text)){     echo "&lt;font color=red&gt;AOPHP Says: PHP Tried To Say '\$text', But I Said No.&lt;/font&gt;&lt;/br&gt;"; }</pre>
▶ MFAOPHP	
▶ phpaspect	<pre>after(): execr(add(\$x,\$y))   execr(div(\$a,\$b)){     echo "&lt;font color=red&gt;I Just did Some Math&lt;/font&gt;&lt;/br&gt;"; }</pre>
▶ Fazit	



## Before Talking

▶ Was ist PHP?

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

▶ Funktionsweise

After Talking

▶ Besonderheiten

▶ AOP mit PHP

N1: 5 | N2: 10

▶ aopinphp

Im About To Add/Sub 5 & 10

▶ Bunny Aspects

Im Going to Increment 5 by 1, then Add it

▶ Seasar

I Just did Some Math

▶ aoPHP

Total of N1 & N2: 16

▶ MFAOPHP

Cant Divide by 0, Returning -1

▶ phpaspect

I Just did Some Math

▶ Fazit

Quotient of N1 & 0: -1



- ▶ Was ist PHP?
  - ▶ Funktionsweise
  - ▶ Besonderheiten
  - ▶ AOP mit PHP
  - ▶ aopinphp
  - ▶ Bunny Aspects
  - ▶ Seasar
  - ▶ aoPHP
  - ▶ MFAOPHP
  - ▶ phaspect
  - ▶ Fazit
- Vorteile
    - Ohne Anpassung der Implementierung bestehender Klassen verwendbar
    - Weaving ohne Aktivierung durch den Entwickler
  - Nachteile
    - Betriebssystemabhängig, nur Linux OS
    - Root Level System Zugriff nötig
    - Läuft nur auf Apache Web Server
    - Benötigt Java JDK ab 1.5



▶ Was ist PHP?

aspectPHP

▶ Funktionsweise

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

▶ Besonderheiten

▶ AOP mit PHP

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

▶ aopinphp

▶ Bunny Aspects

▶ Seasar

▶ aoPHP

▶ MFAOPHP

▶ phaspect

▶ Fazit



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

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

<b>classkit support</b>	<b>enabled</b>
<b>version</b>	<b>0.4</b>



- Vorgehensweise

- ▶ Was ist PHP?
  - Implementieren der mit Aspekten zu versehenen Klassen wie gewöhnlich
- ▶ Funktionsweise
  - Erzeugen von Pointcuts
- ▶ Besonderheiten
  - Hinzufügen von Joinpoints zu den Pointcuts
- ▶ AOP mit PHP
  - Aspekte mit den Pointcuts und zugehöriger Logik als Übergabeparameter anlegen
- ▶ aopinphp
- ▶ Bunny Aspects
- ▶ Seasar
- ▶ aoPHP
- ▶ MFAOPHP
- ▶ phaspect
- ▶ Fazit



	<?PHP
▶ Was ist PHP?	require_once('./JoinPoint.php');
▶ Funktionsweise	require_once('./PointCut.php');
▶ Besonderheiten	require_once('./Aspect.php');
▶ AOP mit PHP	
▶ aopinphp	class HelloWorld {
▶ Bunny Aspects	function sayHello() {
▶ Seasar	echo "Hello World ";
▶ aoPHP	}
▶ MFAOPHP	function anotherHello() {
▶ phpaspect	echo "Hello World ";
▶ Fazit	return "Hello World ";
	}
	}



▶ Was ist PHP?

▶ Funktionsweise

▶ Besonderheiten

▶ AOP mit PHP

▶ aopinphp

▶ Bunny Aspects

▶ Seasar

▶ aoPHP

▶ **MFAOPHP**

▶ phaspect

▶ Fazit

```
$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
";');
```

```
$test2 = new Aspect ($testPointCut, after, 'echo "After $MethodName
";');
```

```
$test3 = new Aspect ($testPointCut, around, '$Return = "New return value of
$MethodName
";');
```

```
$myHelloWorld->sayHello();
```

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

```
?>
```



▶ Was ist PHP?

▶ Funktionsweise

▶ Besonderheiten

▶ AOP mit PHP

▶ aopinphp

▶ Bunny Aspects

▶ Seasar

▶ aoPHP

▶ MFAOPHP

▶ phpaspect

▶ Fazit

- 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);
?>
```



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

- Weaving durch Modifizieren von Klassen zur Laufzeit

- Vorteile

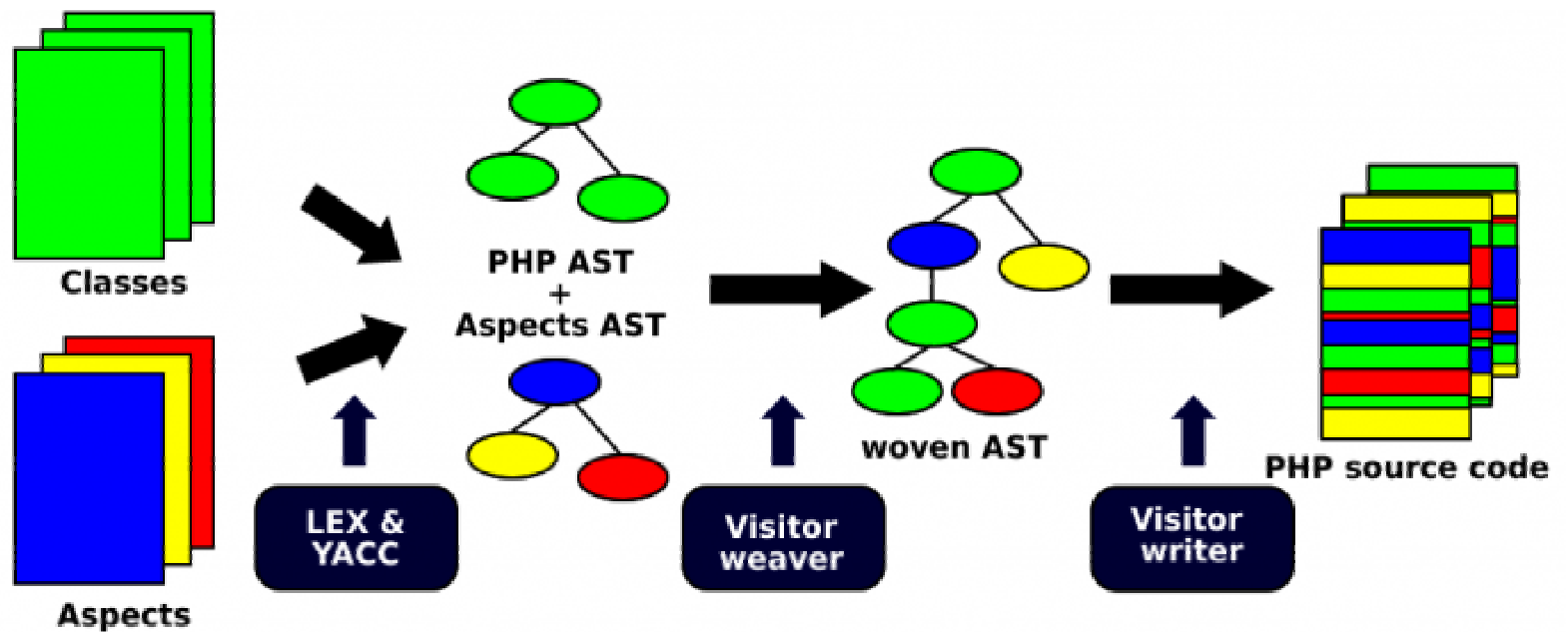
- Kein Übersetzungsvorgang nötig
  - Direkte Ausführung des geschriebenen Codes
- Relativ einfach einsetzbar, da lediglich Classkit installiert werden muss
- Ohne Anpassung der Implementierung bestehender Klassen verwendbar

- Nachteile

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

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

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



LEX / YACC: Compilerbauwerkzeuge    AST: Abstract Syntax Tree



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



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

- Vorgehensweise

- Implementieren der mit Aspekten zu versehenen Klassen wie gewöhnlich
- Definieren der Pointcuts mit der
  - before
  - after
  - und around

auszuführenden Logik in einem Programmmodul mit dem Schlüsselwort **aspect**

- Weaving mit phpaspect-Compiler
  - Aufruf: php Pfad\phpaspect.php Quellpfad Zielpfad



▶ Was ist PHP?	<?php
▶ Funktionsweise	class HelloWorld{
▶ Besonderheiten	public function sayHello(){
▶ AOP mit PHP	echo "Hello World ";
▶ aopinphp	}
▶ Bunny Aspects	}
▶ Seasar	\$myHelloWorld = new HelloWorld();
▶ aoPHP	\$myHelloWorld->sayHello();
▶ MFAOPHP	?>
▶ <b>phpaspect</b>	<?php
▶ Fazit	aspect TraceHello{
	pointcut logSayHello:exec(public HelloWorld->sayHello(0));
	after logSayHello{
	echo "sayHello executed ";
	}
	}
	?>



	(...)
▶ Was ist PHP?	class HelloWorld {
▶ Funktionsweise	public function sayHello() {
▶ Besonderheiten	echo "Hello World ";
▶ AOP mit PHP	\$thisJoinPoint = new
▶ aopinphp	ExecJoinPoint(array('name'=>'logSayHello',
▶ Bunny Aspects	'class'=>'HelloWorld', 'method'=>'sayHello',
▶ Seasar	'arguments'=>'0'), '2', 'HelloWorld.php', array(),
▶ aoPHP	\$this);
▶ MFAOPHP	echo "sayHello executed ";
▶ phpaspect	unset(\$thisJoinPoint);
▶ Fazit	}
	}
	(...)



- ▶ Was ist PHP?
  - ▶ Funktionsweise
  - ▶ Besonderheiten
  - ▶ AOP mit PHP
  - ▶ aopinphp
  - ▶ Bunny Aspects
  - ▶ Seasar
  - ▶ aoPHP
  - ▶ MFAOPHP
  - ▶ **phpaspect**
  - ▶ Fazit
- 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: `pointcut 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;
}
```



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

- 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 Entwicklungsstadium
  - Bugs
  - Kryptische Fehlermeldungen



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

▶ Was ist PHP?

▶ Funktionsweise `<aspect name="Trace">`

▶ Besonderheiten `<pointcut name="HelloWorld" type="call" select="$myHelloWorld->sayHello">`  
`<before>echo "HelloWorld::sayHello() will be executed";</before>`

▶ AOP mit PHP `<after>echo "HelloWorld::sayHello() has been executed";</after>`

▶ aopinphp `</pointcut>`

`</aspect>`

▶ Bunny Aspects

▶ Seasar `<?php`

`require "../libs/aspect.class.php";`

▶ aoPHP `$aspect = new Aspect(__FILE__, realpath('aspects/helloworld.xml'));`

▶ MFAOPHP `$aspect->setClass('HelloWorld', 'classes/helloworld.class.php');`

`$aspect->weave();`

▶ phpaspect `$myHelloWorld = new HelloWorld;`

▶ Fazit `$myHelloWorld->sayHelloWorld();`

`?>`



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

- **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";
?>
```



▶ Was ist PHP?

▶ Funktionsweise

▶ Besonderheiten

▶ AOP mit PHP

▶ aopinphp

▶ Bunny Aspects

▶ Seasar

▶ aoPHP

▶ MFAOPHP

▶ phpaspect

▶ 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

---

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





- ▶ Was ist PHP?
  - aoPHP
    - <http://www.aopHP.net/>
    - <http://sourceforge.net/projects/aopHP/>
- ▶ Funktionsweise
- ▶ Besonderheiten
  - aopHP
    - <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://www.weberdev.com/get_example-4237.html)
  - aspectPHP
    - <http://www.cs.toronto.edu/~yijun/aspectPHP/>
- ▶ AOP mit PHP
- ▶ aopHP
  - Bunny Aspects
    - [http://enterprise.phpmagazine.net/2005/12/aspect\\_oriented\\_programming\\_fo.html](http://enterprise.phpmagazine.net/2005/12/aspect_oriented_programming_fo.html)
    - [http://freshmeat.net/projects/aopHP/?branch\\_id=61075&release\\_id=219924](http://freshmeat.net/projects/aopHP/?branch_id=61075&release_id=219924)
- ▶ Seasar
- ▶ aoPHP
- ▶ MFAOPHP
- ▶ phpaspect
  - 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>
- ▶ Fazit



- ▶ Was ist PHP? • **phpaspect**
  - <http://phpaspect.org/wiki/doku.php?id=phpaspect>
- ▶ Funktionsweise
  - <http://groups.google.com/group/phpaspect>
- ▶ Besonderheiten • **MFAOPHP**
  - <http://www.mfaop.com/>
- ▶ AOP mit PHP • **Seasar**
  - <http://www.seasar.org/en/php5/index.html>
- ▶ aopinphp • **Sonstiges zu AOP in PHP**
  - [http://jaxn.org/blog/category/technology/php/aspect\\_oriented/](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>
- ▶ Bunny Aspects
  - <http://www.sebastian-bergmann.de/AspectPHP/>
  - <http://www.phpcompiler.org/spinoffs/index.html>
- ▶ Seasar
- ▶ aoPHP
- ▶ MFAOPHP
- ▶ phpaspect
- ▶ Fazit



- PHP

- ▶ Was ist PHP? – <http://de.wikipedia.org/wiki/Php>
- ▶ Funktionsweise – <http://www.ister.org/code/article/de/phpbench51.xhtml>
- ▶ Besonderheiten – <http://www.php.net/manual/de/>
- ▶ AOP mit PHP – <http://www.php.net/usage.php>
- ▶ aopinphp – <http://www.php.net/manual/de/language.oop.php>
- ▶ Bunny Aspects – <http://www.php.net/manual/de/language.oop5.php>
- ▶ Seasar – <http://www.apachefriends.org/de/index.html>
- ▶ aoPHP
- ▶ MFAOPHP
- ▶ phaspect
- ▶ Fazit