

# **Annotations in PHP**

Von Dokumentation bis Funktionalität

Thorsten Hallwas, Software Engineer

Florian Sonnenburg, Project Lead



## Thorsten Hallwas

Software Engineer ICANS GmbH  
Dipl.-Inf. (FH)  
Zend Certified Engineer (ZCE)  
[thorsten.hallwas@icans-gmbh.com](mailto:thorsten.hallwas@icans-gmbh.com)



## Florian Sonnenburg

Project Lead ICANS GmbH  
Dipl.-Ing. (FH) Medientechnik  
Zend Certified Engineer (ZCE)  
[florian.sonnenburg@icans-gmbh.com](mailto:florian.sonnenburg@icans-gmbh.com)

- Kommentieren in PHP
- Code Analyse Steuerung
- Unittest Steuerung
- Reflection in PHP
- Einsatz in Symfony2
  - Routen
  - Templates
  - Validierung
  - Absicherung
  - Dependency Injection
  - Dokumente und Tabellen
- Zusammenfassung
  - Vorteile
  - Nachteile
  - Ausblick
- Existierende Annotationparser
- Übersicht der verwendeten Tools und Frameworks

- Code-Dokumentation für Entwickler
- Annotierte Dokumentation ermöglicht automatischen Export
- Tools wie PHPDocumentor, Doxygen, ApiGen erstellen navigierbare API Doku
- Autovervollständigung in modernen IDEs

```
<?php
/**
 * Declares MessageServiceInterface
 *
 * @author Thorsten 'stepo' Hallwas
 * @copyright 2012 ICANS GmbH
 */

/**
 * Interface for the message service.
 */
interface MessageServiceInterface
{
    /**
     * Stores a new message in the database.
     *
     * @param MessageInterface $message
     *
     * @return MessageInterface The stored message.
     *
     * @throws UnableToStoreMessageException
     */
    public function create(MessageInterface $message);
}
```

Sinnvoll bei Nutzung von Bibliotheken Dritter, z.B.:

## Steuern von Mess Detection-Warnungen (phpmd)

- @SuppressWarnings(PHPMD) um alle zu unterdrücken
- Genauerere Angabe der Regel, hier ungenutzte Methodenvariable:  
  - @SuppressWarnings  
    (PHPMD.UnusedFormalParameter)

## Steuern von Codestyle-Warnungen (phpcs)

- Blockweise durch @codingStandardsIgnoreStart und @codingStandardsIgnoreEnd
- Ganze Datei durch @codingStandardsIgnoreFile

```
<?php
namespace Icans\Project\Component\Logging\Flume;

/**
 * @generated
 * @codeCoverageIgnore
 * @codingStandardsIgnoreFile
 */
final class ThriftFlumeRegistry
{
    static public function registerFile($file)
    {
        require_once $file;
    }
}
```

- Annotations in Test-Klassen: Viele Features von PHPUnit durch Annotations kontrollierbar, z.B.:
  - Test-Definitionen (@test, @covers, ...)
  - Steuerung von CodeCoverage (@codeCoverageIgnore)
- Annotations in den zu testenden Klassen
  - Assertions auf Exceptions (@expectedException)
  - Viele weitere unter <http://phpunit.de>

```
<?php
class MessageServiceTest extends \PHPUnit_Framework_TestCase
{
    /**
     * Tests that an exception is thrown if the document manager fails.
     *
     * @test
     * @expectedException \Icans\UnableToStoreMessageException
     */
    public function createException()
    {
        $messageMock = $this->getMock('\Icans\MessageInterface');

        $this->documentManagerMock
            ->expects($this->once())
            ->method('persist')
            ->will($this->throwException(new MongoDBException));

        $this->service->create($messageMock);
    }
}
```

- Reverse-Engineering von PHP Code (Klassen, Interfaces, Funktionen, Methoden)
- Ermöglicht das Auslesen von DocBlocks

```
/**  
 * Controller to show the reflection functions.  
 *  
 * @ignore  
 * @codeCoverageIgnore  
 * @see http://php.net/manual/en/book.reflection.php  
 */  
class ReflectionController extends Controller  
{  
    /**  
     * @var string  
     */  
    protected $reflectionProperty;  
  
    /**  
     * Showing how the Reflection is able to access the Documentation.  
     *  
     * @Route("/reflection/")  
     * @Template()  
     *  
     * @return array  
     */  
    public function reflectionAction()  
    {  
    }  
}
```

getDocComment() liest Dokumentations-Blöcke, z.B. Klasse:

```
$reflectionClass = new \ReflectionClass($this);
$classDocumentation = $reflectionClass->getDocComment();
preg_match('/@see\s+(.*)\n/', $classDocumentation, $seeAnnotation);
```

Methode:

```
$reflectedMethod = $reflectionClass->getMethod('reflectionAction');
$methodDocumentation = $reflectedMethod->getDocComment();
preg_match('/@Route\(.*\)\/', $methodDocumentation, $routeAnnotations);
```

Eigenschaft:

```
$reflectedProperty = $reflectionClass->getProperty('reflectionProperty');
$propertyDocumentation = $reflectedProperty->getDocComment();
preg_match('/@var\s+(.*)\n/', $propertyDocumentation, $typeAnnotation);
```

## Class:

```
/**  
 * Controller to show the reflection functions.  
 * @ignore  
 * @codeCoverageIgnore  
 * @see http://php.net/manual/en/book.reflection.php  
 */
```

<http://php.net/manual/en/book.reflection.php>

## Method

```
/**  
 * Showing how the Reflection is able to access the Documentation.  
 *  
 * @Route("/reflection/")  
 * @Template()  
 *  
 * @return array  
 */
```

### Route Annotation:

```
@Route("/reflection/")
```

## Property:

```
/**  
 * @var string  
 */
```

### Typ:

```
string
```

- Symfony 2 ist als Glue- oder Full-Stack-Framework verwendbar
- Aufteilung in Bundles
- Verwendung von Annotationen, um Funktionalität von Konfiguration zu trennen
- Verwendeter Annotation-Parser: Doctrine 2
- Nutzt Code-Generierung um Konfigurationen in optimierbaren Code umzuwandeln

### Verwendungsbeispiele:

- Routing – FrameworkExtraBundle
- Templates – FrameworkExtraBundle
- Validierung – FrameworkExtraBundle
- Absicherung – JMSecurityExtraBundle
- DependencyInjection – JMSDiExtraBundle
- Doctrine ORM Konfiguration – DoctrineBundle
- Viele weitere verfügbar



Definition der Route direkt an Action im Controller:

```
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
[...]
/**
 * Controller to demonstrate the use of Annotations in Symfony2.
 */
class DemoController extends Controller
{
    /**
     * Displays the newest 5 messages.
     *
     * @Route("/", name="annotationdemo_viewMessageList")
    [...]
    */
    public function viewMessageListAction()
```

Eigenschaft:

```
<?xml version="1.0" encoding="UTF-8" ?>

<routes xmlns="http://symfony.com/schema/routing"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://symfony.com/schema/routing
        http://symfony.com/schema/routing-1.0.xsd">

    <route id="annotationdemo_viewMessageList" pattern="/">
        <default key="_controller">
            IcansAnnotationsAnnotationDemoBundle:Demo:viewMessageList
        </default>
    </route>
</routes>
```

Beide Ansätze werden zu folgendem Code umgewandelt:

```
/**  
 * appprodurlMatcher  
 *  
 * This class has been auto-generated  
 * by the Symfony Routing Component.  
 */  
class appprodurlMatcher extends Symfony\Bundle\FrameworkBundle\Routing\RedirectableurlMatcher  
{  
    /**  
     * Constructor.  
     */  
    public function __construct(RequestContext $context)  
    {  
        $this->context = $context;  
    }  
  
    public function match($pathinfo)  
    {  
        $allow = array();  
        $pathinfo = rawurlencode($pathinfo);  
  
        // annotationdemo_viewMessageList  
        if (rtrim($pathinfo, '/') === '') {  
            if (substr($pathinfo, -1) !== '/') {  
                return $this->redirect($pathinfo . '/', 'annotationdemo_viewMessageList');  
            }  
  
            return array(  
                '_controller' => 'Icans\\[...]\\DemoController::viewMessageListAction',  
                '_route' => 'annotationdemo_viewMessageList',  
            );  
        }  
    }  
}
```

Definition des Templates direkt an Action im Controller:

```
[...]
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;
[...]
 * @Route("/", name="annotationdemo_viewMessageList")
 * @Template()
 *
 * @return array
 */
public function viewMessageListAction()
{
    $messages = $this->messageService->findAll(5, 0);

    return array('messages' => $messages);
}
```

Entspricht @Template("IcansAnnotationsAnnotationDemoBundle:Demo:viewMessageList.html.twig")

Alternativer Aufruf:

```
[...]
 * @Route("/", name="annotationdemo_viewMessageList")
 *
 * @return Response
 */
public function viewMessageListAction()
{
    $messages = $this->messageService->findAll(5, 0);
    return $this->render(
        'IcansAnnotationsAnnotationDemoBundle:Demo:viewMessageList.html.twig',
        array('messages' => $messages)
    );
}
```

- Teil der Form Komponenten
- Annotation der Gewünschten Werte von Eigenschaften der Klasse
- Kombinierbar
- Einfach um eigene Validatoren zu erweitern

```
[...]
use Symfony\Component\Validator\Constraints as Assert;

/**
 * Mongodb document holding a message.
 *
 * @codeCoverageIgnore
 */
class Message implements MessageInterface
{
    /**
     * @Assert\MinLength(5)
     * @Assert\MaxLength(100)
     *
     * @var string
     */
    protected $message;
```

- Absicherung für Parameter, Klassen oder Methoden
- @Secure sichert Controller Actions durch Rollensystem ab
- Teil eines „Interceptor“ Systems durch den der Zugriff in einer Proxy Klasse verhindert werden kann.

```
[...]  
  
use JMS\SecurityExtraBundle\Annotation\Secure;  
  
[...]  
  
/**  
 * Shows the create new message form.  
 *  
 * @Secure(roles="ROLE_USER")  
 * @Route("/create/", name="annotationdemo_createMessage")  
 * @Template()  
 *  
 * @param Request      $request  
 * @param MessageService $messageService  
 * @return array  
 */  
public function createNewMessageAction(Request $request)
```

Löst Abhängigkeiten ohne nötige Konfiguration auf.

```
use JMS\DiExtraBundle\Annotation\Inject;

/**
 * Controller to demonstrate the use of Annotations in Symfony2.
 */
class DemoController extends Controller
{
    /**
     * @Inject("icans.annotations.annotationdemo.messageservice")
     * @var MessageService
     */
    protected $messageService;
```

Erspart folgenden Code:

```
/**
 * Displays the newest 5 messages.
 */
public function viewMessageListAction()
{
    $messageService = $this->get('icans.annotations.annotationdemo.messageservice');
    [...]
}
```

Definition des Message-Services für die Dependency Injection per XML Konfigurations-Datei:

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

<container xmlns="http://symfony.com/schema/dic/services"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://symfony.com/schema/dic/services
        http://symfony.com/schema/dic/services/services-1.0.xsd">

    <services>
        <service id="icans.annotations.annotationdemo.messageservice"
            class="Icans\Annotations\AnnotationDemoBundle\Service\MessageService">
            <argument type="service" id="doctrine.odm.mongodb.document_manager" />
        </service>

        <service id="icans.annotations.annotationdemo.messageform"
            class="Icans\Annotations\AnnotationDemoBundle\Form\Type\MessageType">
            <tag name="form.type" alias="icans_annotations_annotationdemo_message" />
        </service>
    </services>
</container>
```

Definition des Message Services für die Dependency Injection per Annotation:

```
use JMS\DiExtraBundle\Annotation\Inject;
use JMS\DiExtraBundle\Annotation\Service;
use JMS\DiExtraBundle\Annotation\InjectParams;

/**
 * Service to store messages into the MongoDB.
 * @Service("icans.annotations.annotationdemo.messageservice", public=true)
 */
class MessageService implements MessageServiceInterface
{
    /**
     * @var DocumentManager
     */
    protected $documentManager;

    /**
     * Constructor.
     * @param DocumentManager $documentManager
     * @InjectParams({
     *     "documentManager" = @Inject("doctrine.odm.mongodb.document_manager")
     * })
     */
    public function __construct(DocumentManager $documentManager)
    {
        $this->documentManager = $documentManager;
    }
}
```

- Doctrine unterstützt mit ORM und ODM eine breite Anzahl von Datenbanken.
- Mit Annotationen können Modelklassen einfach an eine Persistenz geknüpft werden
- Notwendige Datentyp für die Datenbank können validiert und vorbereitet werden.

```
use Doctrine\ODM\MongoDB\Mapping\Annotations as MongoDB;
use Symfony\Component\Validator\Constraints as Assert;

/**
 * MongoDB document holding a message.
 *
 * @MongoDB\Document
 */
class Message implements MessageInterface
{
    /**
     * @MongoDB\Id(strategy="auto")
     * @var \MongoId
     */
    protected $id;

    /**
     * @MongoDB\String
     * @var string
     */
    protected $message;

    /**
     * @MongoDB\ReferenceOne(targetDocument="Icans\Annotations\UserBundle\Document\User")
     * @var UserInterface
     */
    protected $author;
```

## Vorteile

- Implementation enthält nur noch reine Logik ohne Konfiguration
- Schlankerer Code
- Weniger Code-Duplizierung
- Erhöhte Lesbarkeit
- Leichtere Testbarkeit
- Konfiguration am Code

## Nachteile

- Debugging wird erschwert durch generierten Code
- Auflösung von Code-Abhängigkeiten in Tests
- Reduzierte Portabilität von Code
- Noch keine Auto vervollständigung in IDEs

### Ausblick

- Viele weitere Annotationen bereits im Symfony Umfeld vorhanden.
- Andere Frameworks mit ähnlichen Ansätzen vorhanden
- Auch bei Java waren Annotationen erst später Teil der Sprache
- IDEs unterstützen immer mehr bei der Auto vervollständigung der Annotationen

Doctrine2



Addendum von Google



Nette Framework



**Symfony 2** – Gluestack Framework auch als Fullstack verwendbar

**PHPUnit** – Testsuite mit Code-Coverage und im Test dynamisch erstellbarer Mocks

**PHP Code Sniffer** – Code Analyse Tool um Stil und Richtlinien zu testen.

**PHP Mess Detector** – Code Analyse Tool um Abhängigkeiten im Code darzustellen, Komplexität zu errechnen und Problemstellen im Code aufzuzeigen

**ICANS Annotation Demo Application** – Zu finden unter github:

<https://github.com/ICANS/AnnotationDemo>





ICANS GmbH

<http://www.icans-gmbh.com>



etruvian

<http://www.etruvian.com>

**Thorsten Hallwas**

Software Engineer ICANS GmbH

Dipl.-Inf. (FH)

Zend Certified Engineer (ZCE)

[thorsten.hallwas@icans-gmbh.com](mailto:thorsten.hallwas@icans-gmbh.com)

**Florian Sonnenburg**

Project Lead ICANS GmbH

Dipl.-Ing. (FH)

Zend Certified Engineer (ZCE)

[Florian.sonnenburg@icans-gmbh.com](mailto:Florian.sonnenburg@icans-gmbh.com)

Aktuelle Informationen gibt es hier:



<http://www.facebook.com/ICANS.GmbH>



<http://devblog.icans-gmbh.com/>