Vom lokalen Build zum Deployment

International PHP Conference

Manuel Pichler

12.10.2011

Über mich

- Diplominformatiker

Über mich

- Diplominformatiker
- Mehr als 10 Jahre Erfahrung im PHP-Umfeld

Über mich

- Diplominformatiker
- Mehr als 10 Jahre Erfahrung im PHP-Umfeld
- Autor hinter einer Reihe von QA-Tools

Über mich

- Diplominformatiker
- Mehr als 10 Jahre Erfahrung im PHP-Umfeld
- Autor hinter einer Reihe von QA-Tools

Mitgründer von

Über mich

- Diplominformatiker
- Mehr als 10 Jahre Erfahrung im PHP-Umfeld
- Autor hinter einer Reihe von QA-Tools

Mitgründer von

We help people to create high quality PHP applications.

Über mich

- Diplominformatiker
- Mehr als 10 Jahre Erfahrung im PHP-Umfeld
- Autor hinter einer Reihe von QA-Tools

Mitgründer von

We help people to create high quality PHP applications.

`http://qafoo.com`

Outline

Motivation

Die Idee

Anforderungen

Features

Roadmap

Fazit

Dezember 2009

- Ausgangssituation

Dezember 2009

- Ausgangssituation
  - Diverse Open-Source-Projekte

Dezember 2009

- Ausgangssituation
  - Diverse Open-Source-Projekte
    - PHP_Depend

Dezember 2009

- Ausgangssituation
  - Diverse Open-Source-Projekte
    - PHP_Depend
    - PHPMD

Dezember 2009

- Ausgangssituation
  - Diverse Open-Source-Projekte
    - PHP_Depend
    - PHPMD
    - phpUnderControl

Dezember 2009

- Ausgangssituation
  - Diverse Open-Source-Projekte
    - PHP_Depend
    - PHPMD
    - phpUnderControl
    - staticReflection

Dezember 2009

- Ausgangssituation
  - Diverse Open-Source-Projekte
    - PHP_Depend
    - PHPMD
    - phpUnderControl
    - staticReflection
    - ...

Dezember 2009

- Ausgangssituation
  - Diverse Open-Source-Projekte
    - PHP_Depend
    - PHPMD
    - phpUnderControl
    - staticReflection
    - ...
  - **Und viel zu wenig Zeit**

Dezember 2009

- Ausgangssituation
  - Diverse Open-Source-Projekte
    - PHP_Depend
    - PHPMD
    - phpUnderControl
    - staticReflection
    - ...
  - **Und viel zu wenig Zeit :-[**

Warum?

- Einheitliche Strukturen?

pdepend /

name	age
PHP/	December 09, 2009
docs/	April 04, 2008
scripts/	August 03, 2009
tests/	November 24, 2009
.gitignore	August 03, 2009
CHANGELOG	November 15, 2009
LICENSE	December 09, 2009
build.nightly.xml	August 02, 2009
build.xml	July 24, 2009
catalog.xml	January 25, 2009
package.xml	November 15, 2009
pdepend.bat	January 04, 2009

Warum?

- Einheitliche Strukturen? Naja

pdepend /

name	age
PHP/	Dece
docs/	Apri
scripts/	Augu
tests/	Nove
.gitignore	Augu
CHANGELOG	Nove
LICENSE	Dece
build.nightly.xml	Augu
build.xml	July
catalog.xml	Janu
package.xml	Nove
pdepend.bat	Janu

phpmd /

name	age
conf/	December 11, 2009
rulesets/	December 27, 2009
source/	January 03, 2010
test/	January 03, 2010
.gitignore	December 29, 2009
CHANGELOG	January 03, 2010
LICENSE	January 03, 2010
build.properties	December 24, 2009
build.xml	December 11, 2009
package.xml	December 29, 2009
phpmd.bat	December 13, 2009
phpmd.php	December 11, 2009

Warum?

- Einheitliche Strukturen? Naja, Vielleicht

pdepend /

name	age
PHP/	Dece
docs/	Apri
scripts/	Augu
tests/	Nove
.gitignore	Augu
CHANGELOG	Nove
LICENSE	Dece
build.nightly.xml	Augu
build.xml	July
catalog.xml	Janu
package.xml	Nove
pdepend.bat	Janu

phpmd /

name	age
conf/	Dece
rulesets/	Dece
source/	Janu
test/	Janu
.gitignore	Dece
CHANGELOG	Janu
LICENSE	Janu
build.properties	Dece
build.xml	Dece
package.xml	Dece
phpmd.bat	Dece
phpmd.php	Dece

phpUnderControl /

name	age
bin/	September 02, 2009
data/	November 19, 2009
docs/	January 04, 2009
java/	January 04, 2009
lib/	February 28, 2009
src/	November 19, 2009
tests/	August 30, 2009
CHANGELOG	September 08, 2008
LICENSE	March 26, 2008
build.xml	January 04, 2009
package.xml	September 19, 2009

Warum?

- Einheitliche Strukturen? Naja, Vielleicht doch nicht ;-]

pdepend /

name	age
PHP/	Dece
docs/	Apri
scripts/	Augu
tests/	Nove
.gitignore	Augu
CHANGELOG	Nove
LICENSE	Dece
build.nightly.xml	Augu
build.xml	July
catalog.xml	Janu
package.xml	Nove
pdepend.bat	Janu

phpmd /

name	age
conf/	Dece
rulesets/	Dece
source/	Janu
test/	Janu
.gitignore	Dece
CHANGELOG	Janu
LICENSE	Janu
build.properties	Dece
build.xml	Dece
package.xml	Dece
phpmd.bat	Dece
phpmd.php	Dece

phpUnderControl /

name	age
bin/	Septe
data/	Novem
docs/	Janua
java/	Janua
lib/	Febru
src/	Novem
tests/	August 30, 2009
CHANGELOG	September 08, 2008
LICENSE	March 26, 2008
build.xml	January 04, 2009
package.xml	September 19, 2009

staticReflection /

name	age
..	
source/	October 23, 2009
test/	October 23, 2009
.gitignore	October 19, 2009
build.xml	October 19, 2009
test.php	October 19, 2009

Build-Infrastruktur

- Jedes Projekt hatte ein individuelles Set an Skripten

Build-Infrastruktur

- Jedes Projekt hatte ein individuelles Set an Skripten
  - Aktualisiere die PEAR package.xml Datei

Build-Infrastruktur

- Jedes Projekt hatte ein individuelles Set an Skripten
  - Aktualisiere die PEAR package.xml Datei
  - Erzeuge ein PEAR- oder Phar-Archiv

Build-Infrastruktur

- Jedes Projekt hatte ein individuelles Set an Skripten
  - Aktualisiere die PEAR package.xml Datei
  - Erzeuge ein PEAR- oder Phar-Archiv
  - Generiere weitere Projekt-Artefakte

Build-Infrastruktur

- Jedes Projekt hatte ein individuelles Set an Skripten
  - Aktualisiere die PEAR package.xml Datei
  - Erzeuge ein PEAR- oder Phar-Archiv
  - Generiere weitere Projekt-Artefakte
- Zusätzlich ein eigenes Ant build.xml Skript

Build-Infrastruktur

- Jedes Projekt hatte ein individuelles Set an Skripten
  - Aktualisiere die PEAR package.xml Datei
  - Erzeuge ein PEAR- oder Phar-Archiv
  - Generiere weitere Projekt-Artefakte
- Zusätzlich ein eigenes Ant build.xml Skript
  - Führe die Tests aus

Build-Infrastruktur

- Jedes Projekt hatte ein individuelles Set an Skripten
  - Aktualisiere die PEAR package.xml Datei
  - Erzeuge ein PEAR- oder Phar-Archiv
  - Generiere weitere Projekt-Artefakte
- Zusätzlich ein eigenes Ant build.xml Skript
  - Führe die Tests aus
  - Prüfe den Coding-Standard

Build-Infrastruktur

- Jedes Projekt hatte ein individuelles Set an Skripten
  - Aktualisiere die PEAR package.xml Datei
  - Erzeuge ein PEAR- oder Phar-Archiv
  - Generiere weitere Projekt-Artefakte
- Zusätzlich ein eigenes Ant build.xml Skript
  - Führe die Tests aus
  - Prüfe den Coding-Standard
  - Sammle Softwaremetriken

Build-Infrastruktur

- Jedes Projekt hatte ein individuelles Set an Skripten
  - Aktualisiere die PEAR package.xml Datei
  - Erzeuge ein PEAR- oder Phar-Archiv
  - Generiere weitere Projekt-Artefakte
- Zusätzlich ein eigenes Ant build.xml Skript
  - Führe die Tests aus
  - Prüfe den Coding-Standard
  - Sammle Softwaremetriken
  - Erstelle einen Release

Build-Infrastruktur

- Jedes Projekt hatte ein individuelles Set an Skripten
  - Aktualisiere die PEAR package.xml Datei
  - Erzeuge ein PEAR- oder Phar-Archiv
  - Generiere weitere Projekt-Artefakte
- Zusätzlich ein eigenes Ant build.xml Skript
  - Führe die Tests aus
  - Prüfe den Coding-Standard
  - Sammle Softwaremetriken
  - Erstelle einen Release
  - ...

Probleme

- Unterschiedliche Kommandos

Probleme

- Unterschiedliche Kommandos
- Unterschiedliche Workflows

Probleme

- Unterschiedliche Kommandos
- Unterschiedliche Workflows
- Zeitaufwand beim Projektwechsel

Probleme

- Unterschiedliche Kommandos
- Unterschiedliche Workflows
- Zeitaufwand beim Projektwechsel
- Wartungsaufwand der Buildumgebung

Probleme

- Unterschiedliche Kommandos
- Unterschiedliche Workflows
- Zeitaufwand beim Projektwechsel
- Wartungsaufwand der Buildumgebung

DRY!?

Probleme

- Unterschiedliche Kommandos
- Unterschiedliche Workflows
- Zeitaufwand beim Projektwechsel
- Wartungsaufwand der Buildumgebung

DRY!?

Don't Repeat Yourself

Outline

Motivation

Die Idee

Anforderungen

Features

Roadmap

Fazit

Software Lifecycle

- Eigentlich besteht Lebenszyklus jeder Software immer aus den gleichen Bestandteilen

Software Lifecycle

- Eigentlich besteht Lebenszyklus jeder Software immer aus den gleichen Bestandteilen
  - Build

Software Lifecycle

- Eigentlich besteht Lebenszyklus jeder Software immer aus den gleichen Bestandteilen
  - Build
  - Testing

Software Lifecycle

- Eigentlich besteht Lebenszyklus jeder Software immer aus den gleichen Bestandteilen
  - Build
  - Testing
  - Dokumentation

Software Lifecycle

- Eigentlich besteht Lebenszyklus jeder Software immer aus den gleichen Bestandteilen
  - Build
  - Testing
  - Dokumentation
  - Qualitätssicherung

Software Lifecycle

- Eigentlich besteht Lebenszyklus jeder Software immer aus den gleichen Bestandteilen
  - Build
  - Testing
  - Dokumentation
  - Qualitätssicherung
  - Deployment

Software Lifecycle

- Eigentlich besteht Lebenszyklus jeder Software immer aus den gleichen Bestandteilen
  - Build
  - Testing
  - Dokumentation
  - Qualitätssicherung
  - Deployment
  - Dependency-Management

Convention over Configuration

- Andere machen es vor

Convention over Configuration

- Andere machen es vor
  - PEAR

Convention over Configuration

- Andere machen es vor
  - PEAR
  - Zend-Framework

Convention over Configuration

- Andere machen es vor
  - PEAR
  - Zend-Framework
  - PSR-0

Convention over Configuration

- Andere machen es vor
  - PEAR
  - Zend-Framework
  - PSR-0
- Warum sollte dies nicht auch für eine Buildumgebung gelten

Convention over Configuration

- Andere machen es vor
  - PEAR
  - Zend-Framework
  - PSR-0
- Warum sollte dies nicht auch für eine Buildumgebung gelten
  - Buildprozess

Convention over Configuration

- Andere machen es vor
  - PEAR
  - Zend-Framework
  - PSR-0
- Warum sollte dies nicht auch für eine Buildumgebung gelten
  - Buildprozess
  - Verzeichnisstruktur

Convention over Configuration

- Andere machen es vor
  - PEAR
  - Zend-Framework
  - PSR-0
- Warum sollte dies nicht auch für eine Buildumgebung gelten
  - Buildprozess
  - Verzeichnisstruktur
  - Versionierung

Convention over Configuration

- Andere machen es vor
  - PEAR
  - Zend-Framework
  - PSR-0
- Warum sollte dies nicht auch für eine Buildumgebung gelten
  - Buildprozess
  - Verzeichnisstruktur
  - Versionierung
  - Namensgebung

Convention over Configuration

- Andere machen es vor
  - PEAR
  - Zend-Framework
  - PSR-0
- Warum sollte dies nicht auch für eine Buildumgebung gelten
  - Buildprozess
  - Verzeichnisstruktur
  - Versionierung
  - Namensgebung
  - Verteilung / Distribution

Evaluiere existierende Lösungen

- Viel Recherche...

Evaluiere existierende Lösungen

- Viel Recherche...
  - ...brachte fast immer das gleiche Ergebnis

Evaluiere existierende Lösungen

- Viel Recherche...
  - ...brachte fast immer das gleiche Ergebnis
    - **Apache Maven**

Evaluiere existierende Lösungen

- Viel Recherche...
  - ...brachte fast immer das gleiche Ergebnis
    - **Apache Maven**

Genauer

Evaluiere existierende Lösungen

- Viel Recherche...
  - ...brachte fast immer das gleiche Ergebnis
    - **Apache Maven**

Genauer

die Konzepte und Konventionen

Apache Maven

- Formalisiert den Buildprozess vollständig

Apache Maven

- Formalisiert den Buildprozess vollständig
- Setzt sehr stark auf Konventionen

Apache Maven

- Formalisiert den Buildprozess vollständig
- Setzt sehr stark auf Konventionen
- Definiert durchdachte Strukturen

Apache Maven

- Formalisiert den Buildprozess vollständig
- Setzt sehr stark auf Konventionen
- Definiert durchdachte Strukturen
- Modulare und Erweiterbare Architektur

Apache Maven

- Formalisiert den Buildprozess vollständig
- Setzt sehr stark auf Konventionen
- Definiert durchdachte Strukturen
- Modulare und Erweiterbare Architektur
- War sehr Java fokussiert

Apache Maven

- Formalisiert den Buildprozess vollständig
- Setzt sehr stark auf Konventionen
- Definiert durchdachte Strukturen
- Modulare und Erweiterbare Architektur
- War sehr Java fokussiert

War???

Apache Maven

- Formalisiert den Buildprozess vollständig
- Setzt sehr stark auf Konventionen
- Definiert durchdachte Strukturen
- Modulare und Erweiterbare Architektur
- War sehr Java fokussiert

War???

Heute gibt es das **PHP for Maven** Plugin

Apache Maven

- Formalisiert den Buildprozess vollständig
- Setzt sehr stark auf Konventionen
- Definiert durchdachte Strukturen
- Modulare und Erweiterbare Architektur
- War sehr Java fokussiert

War???

Heute gibt es das **PHP for Maven** Plugin

Scheint mittlerweile aber leider unmaintained

Outline

Motivation

Die Idee

Anforderungen

Features

Roadmap

Fazit

Buildwerkzeug

- Apache Ant als Basis

Buildwerkzeug

- Apache Ant als Basis

Warum Ant und nicht Phing, Pake, ...?

Buildwerkzeug

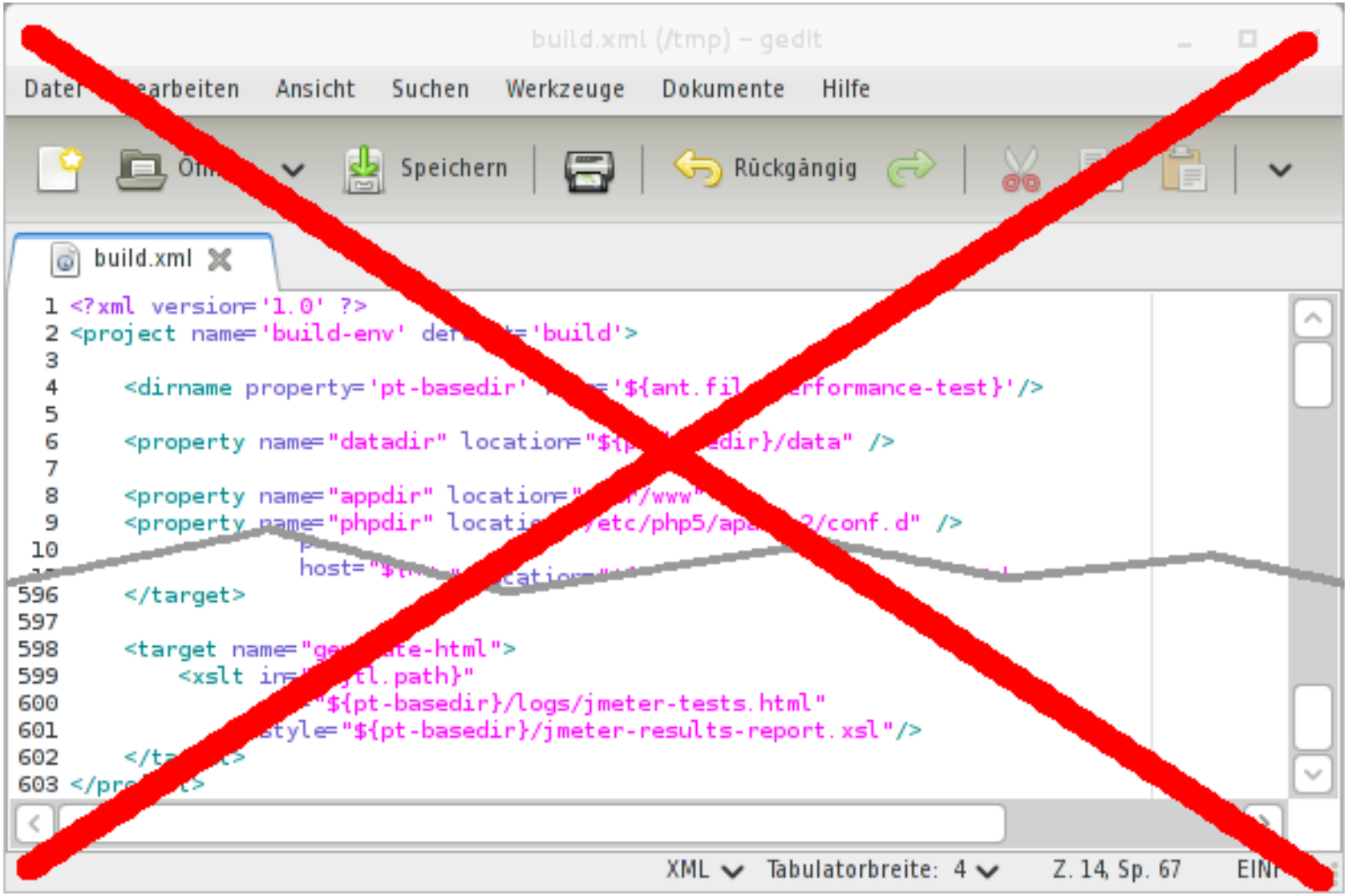
- Apache Ant als Basis

Warum Ant und nicht Phing, Pake, ...?

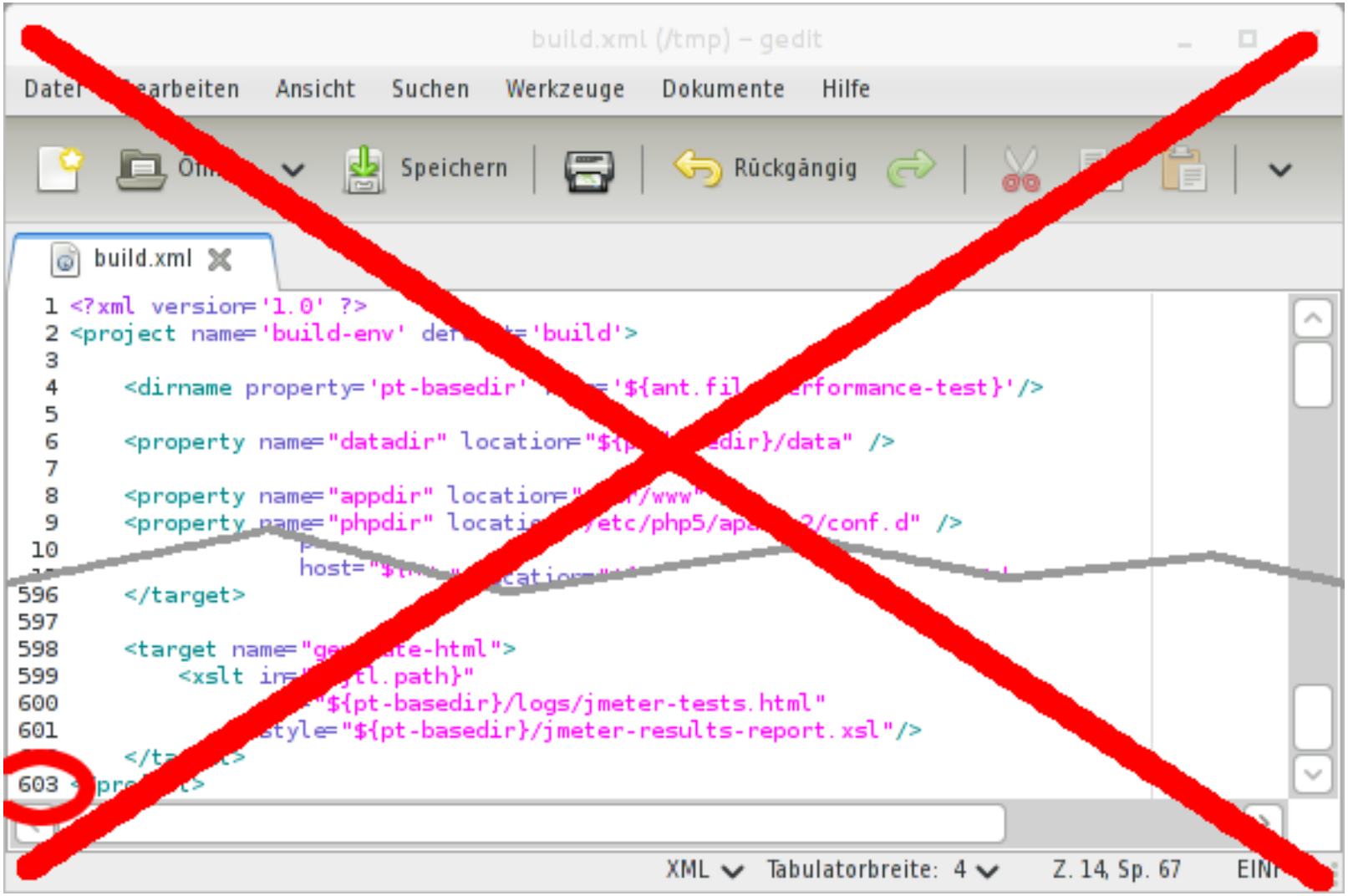
Es gibt keinen wirklichen Grund

Minimale Konfiguration

Minimale Konfiguration



Minimale Konfiguration



Minimale Konfiguration

- `build.xml`

Minimale Konfiguration

- build.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="PHP_Depend" basedir=".">
  <property file="build.properties" />

  <import file="setup/src/main/xml/base.xml" />
</project>
```

Minimale Konfiguration

- build.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="PHP Depend" basedir=".">
  <property file="build.properties" />

  <import file="setup/src/main/xml/base.xml" />
</project>
```

- build.properties

Minimale Konfiguration

- build.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project name="PHP Depend" basedir=".">
  <property file="build.properties" />

  <import file="setup/src/main/xml/base.xml" />
</project>
```

- build.properties

```
project.uri       = pdepend.org
project.name      = pdepend
project.version   = 0.11.0
project.stability = stable
```

Fest definierte Buildschritte

- initialize

Fest definierte Buildschritte

- initialize
- compile

Fest definierte Buildschritte

- initialize
- compile
- test

Fest definierte Buildschritte

- initialize
- compile
- test
- package

Fest definierte Buildschritte

- initialize
- compile
- test
- package
- integration-test

Fest definierte Buildschritte

- initialize
- compile
- test
- package
- integration-test
- verify

Fest definierte Buildschritte

- initialize
- compile
- test
- package
- integration-test
- verify
- install

Fest definierte Buildschritte

- initialize
- compile
- test
- package
- integration-test
- verify
- install
- deploy

Modulare Architektur

- Buildschritte und Abfolge geben die Grundstruktur vor

Modulare Architektur

- Buildschritte und Abfolge geben die Grundstruktur vor
- Jeder Buildschritt soll non-invasiv erweiterbar sein

Modulare Architektur

- Buildschritte und Abfolge geben die Grundstruktur vor
- Jeder Buildschritt soll non-invasiv erweiterbar sein
  - Codegeneratoren

Modulare Architektur

- Buildschritte und Abfolge geben die Grundstruktur vor
- Jeder Buildschritt soll non-invasiv erweiterbar sein
  - Codegeneratoren
  - Testwerkzeuge

Modulare Architektur

- Buildschritte und Abfolge geben die Grundstruktur vor
- Jeder Buildschritt soll non-invasiv erweiterbar sein
  - Codegeneratoren
  - Testwerkzeuge
  - Distributionswerkzeuge

Modulare Architektur

- Buildschritte und Abfolge geben die Grundstruktur vor
- Jeder Buildschritt soll non-invasiv erweiterbar sein
  - Codegeneratoren
  - Testwerkzeuge
  - Distributionswerkzeuge
  - QA-Tools

Modulare Architektur

- Buildschritte und Abfolge geben die Grundstruktur vor
- Jeder Buildschritt soll non-invasiv erweiterbar sein
  - Codegeneratoren
  - Testwerkzeuge
  - Distributionswerkzeuge
  - QA-Tools
  - Datenbank

Modulare Architektur

- Buildschritte und Abfolge geben die Grundstruktur vor
- Jeder Buildschritt soll non-invasiv erweiterbar sein
  - Codegeneratoren
  - Testwerkzeuge
  - Distributionswerkzeuge
  - QA-Tools
  - Datenbank
  - ...

Outline

Motivation

Die Idee

Anforderungen

Features

Roadmap

Fazit

Standardisierte Verzeichnisstruktur

- `build/`
- `src/`

Standardisierte Verzeichnisstruktur

- `build/`
  - `dist/`
  - `logs/`
- `src/`

Standardisierte Verzeichnisstruktur

- build/
  - dist/
  - logs/
- src/
  - bin/
  - conf/
  - main/
  - test/

Standardisierte Verzeichnisstruktur

- build/
  - dist/
  - logs/
- src/
  - bin/
  - conf/
  - main/
    - js/
    - php/
    - resources/
  - test/

Standardisierte Verzeichnisstruktur

- build/
  - dist/
  - logs/
- src/
  - bin/
  - conf/
  - main/
    - js/
    - php/
    - resources/
  - test/
    - js/
    - php/
    - resources/

Environments I

- Es existiert immer ein Standard-Environment
- ...

```
1 ...
2 db.hostname = localhost
3 db.userid   = root
4 db.password =
5 db.name     = dev_db
```

Environments I

- Es existiert immer ein Standard-Environment
- ...

```
db.hostname = localhost
db.userid   = root
db.password =
db.name     = dev_db
```

- Für das Standard-Environment ist lediglich der Buildschritt anzugeben
- `$ ant deploy`

Environments II

- Es können frei weitere Environments hinzugefügt werden

Environments II

- Es können frei weitere Environments hinzugefügt werden
- Jede Variable kann für das Environment redefiniert werden
- ...

```
1 ...
2 db.testing.hostname = db01.test.intern.example.com
3 db.testing.userid   = user
4 db.testing.password = password
5 db.testing.name     = testing_db
```

Environments II

- Es können frei weitere Environments hinzugefügt werden
- Jede Variable kann für das Environment redefiniert werden
- ...

```
db.testing.hostname = db01.test.intern.example.com
db.testing.userid   = user
db.testing.password = password
db.testing.name     = testing_db
```

- Über die Option `commons.env` lässt sich zusätzlich das Environment wählen
- `$ ant -Dcommons.env=testing deploy`

Unabhängige Buildumgebung

- Ein Build der Software sollte möglich sein, ohne zuvor $X Abhängigkeiten installieren zu müssen

Unabhängige Buildumgebung

- Ein Build der Software sollte möglich sein, ohne zuvor $X Abhängigkeiten installieren zu müssen
  - PEAR ist fester Bestandteil der Buildumgebung

Unabhängige Buildumgebung

- Ein Build der Software sollte möglich sein, ohne zuvor $X Abhängigkeiten installieren zu müssen
  - PEAR ist fester Bestandteil der Buildumgebung
    - Alle weiteren Pakete werden erst bei Bedarf installiert

Unabhängige Buildumgebung

- Ein Build der Software sollte möglich sein, ohne zuvor $X Abhängigkeiten installieren zu müssen
  - PEAR ist fester Bestandteil der Buildumgebung
    - Alle weiteren Pakete werden erst bei Bedarf installiert
  - Kein händisches installieren von Paketen
    - `$ pear channel-discover pear.pdepend.org`
    - `$ pear install --alldeps pdepend/php_depend`

Unabhängige Buildumgebung

- Ein Build der Software sollte möglich sein, ohne zuvor $X Abhängigkeiten installieren zu müssen
  - PEAR ist fester Bestandteil der Buildumgebung
    - Alle weiteren Pakete werden erst bei Bedarf installiert
  - ~~Kein händisches installieren von Paketen~~
    - ~~`$ pear channel-discover pear.pdepend.org`~~
    - ~~`$ pear install --alldeps pdepend/php_depend`~~

Unabhängige Buildumgebung

- Ein Build der Software sollte möglich sein, ohne zuvor $X Abhängigkeiten installieren zu müssen
  - PEAR ist fester Bestandteil der Buildumgebung
    - Alle weiteren Pakete werden erst bei Bedarf installiert
  - ~~Kein händisches installieren von Paketen~~
    - ~~`$ pear channel-discover pear.pdepend.org`~~
    - ~~`$ pear install --alldeps pdepend/php_depend`~~
  - Dies erlaubt fest definierte Tool-Versionen je Projekt

Unabhängige Buildumgebung

- Ein Build der Software sollte möglich sein, ohne zuvor $X Abhängigkeiten installieren zu müssen
  - PEAR ist fester Bestandteil der Buildumgebung
    - Alle weiteren Pakete werden erst bei Bedarf installiert
  - ~~Kein händisches installieren von Paketen~~
    - ~~`$ pear channel-discover pear.pdepend.org`~~
    - ~~`$ pear install --alldeps pdepend/php_depend`~~
  - Dies erlaubt fest definierte Tool-Versionen je Projekt
    - `phpunit.package.name = phpunit-3.5.2`

Unabhängige Buildumgebung

- Ein Build der Software sollte möglich sein, ohne zuvor $X Abhängigkeiten installieren zu müssen
  - PEAR ist fester Bestandteil der Buildumgebung
    - Alle weiteren Pakete werden erst bei Bedarf installiert
  - ~~Kein händisches installieren von Paketen~~
    - ~~`$ pear channel-discover pear.pdepend.org`~~
    - ~~`$ pear install --alldeps pdepend/php_depend`~~
  - Dies erlaubt fest definierte Tool-Versionen je Projekt
    - `phpunit.package.name = phpunit-3.5.2`
    - `codesniffer.package.name = php_codesniffer-1.2.0`

Buildschritte I

- initialize

Buildschritte I

- initialize
  - Build Verzeichnisse erstellen

Buildschritte I

- initialize
  - Build Verzeichnisse erstellen
  - Update aus der Versionskontrolle

Buildschritte I

- initialize
  - Build Verzeichnisse erstellen
  - Update aus der Versionskontrolle
- compile

Buildschritte I

- initialize
  - Build Verzeichnisse erstellen
  - Update aus der Versionskontrolle
- compile
  - `php -l` auf Quelltext und Tests

Buildschritte I

- initialize
  - Build Verzeichnisse erstellen
  - Update aus der Versionskontrolle
- compile
  - `php -l` auf Quelltext und Tests
  - Nur geänderte Dateien werden geprüft

Buildschritte I

- initialize
  - Build Verzeichnisse erstellen
  - Update aus der Versionskontrolle
- compile
  - `php -l` auf Quelltext und Tests
  - Nur geänderte Dateien werden geprüft
- test

Buildschritte I

- initialize
  - Build Verzeichnisse erstellen
  - Update aus der Versionskontrolle
- compile
  - `php -l` auf Quelltext und Tests
  - Nur geänderte Dateien werden geprüft
- test
  - Ausführen der Unittests

Buildschritte I

- initialize
  - Build Verzeichnisse erstellen
  - Update aus der Versionskontrolle
- compile
  - `php -l` auf Quelltext und Tests
  - Nur geänderte Dateien werden geprüft
- test
  - Ausführen der Unittests
  - `@covers unittest` Annotation

Buildschritte I

- initialize
  - Build Verzeichnisse erstellen
  - Update aus der Versionskontrolle
- compile
  - `php -l` auf Quelltext und Tests
  - Nur geänderte Dateien werden geprüft
- test
  - Ausführen der Unittests
  - `@covers unittest` Annotation
- package
  - Zip-Archiv

Buildschritte I

- initialize
  - Build Verzeichnisse erstellen
  - Update aus der Versionskontrolle
- compile
  - `php -l` auf Quelltext und Tests
  - Nur geänderte Dateien werden geprüft
- test
  - Ausführen der Unittests
  - `@covers unittest` Annotation
- package
  - Zip-Archiv
  - Optional PEAR-Archiv

Buildschritte I

- initialize
  - Build Verzeichnisse erstellen
  - Update aus der Versionskontrolle
- compile
  - `php -l` auf Quelltext und Tests
  - Nur geänderte Dateien werden geprüft
- test
  - Ausführen der Unittests
  - `@covers unittest` Annotation
- package
  - Zip-Archiv
  - Optional PEAR-Archiv
  - Optional Phar-Archiv

Buildschritt: integration-test

- `@covers integrationtest` Annotation

Buildschritt: integration-test

- `@covers integrationtest` Annotation
- Ausführen der Integrationstests

Buildschritt: integration-test

- `@covers integrationtest` Annotation
- Ausführen der Integrationstests
  - In der Dev-Umgebung

Buildschritt: integration-test

- `@covers integrationtest` Annotation
- Ausführen der Integrationstests
  - In der Dev-Umgebung
  - Gegen die Quellen im Zip-Archiv

Buildschritt: integration-test

- `@covers integrationtest` Annotation
- Ausführen der Integrationstests
  - In der Dev-Umgebung
  - Gegen die Quellen im Zip-Archiv
  - Optional gegen das Phar-Archiv

Buildschritt: integration-test

- `@covers integrationtest` Annotation
- Ausführen der Integrationstests
  - In der Dev-Umgebung
  - Gegen die Quellen im Zip-Archiv
  - Optional gegen das Phar-Archiv
  - Optional gegen das PEAR-Archiv

Buildschritt: integration-test

- `@covers integrationtest` Annotation
- Ausführen der Integrationstests
  - In der Dev-Umgebung
  - Gegen die Quellen im Zip-Archiv
  - Optional gegen das Phar-Archiv
  - Optional gegen das PEAR-Archiv

Exkurs lokales PEAR

Buildschritt: integration-test

- `@covers integrationtest` Annotation
- Ausführen der Integrationstests
  - In der Dev-Umgebung
  - Gegen die Quellen im Zip-Archiv
  - Optional gegen das Phar-Archiv
  - Optional gegen das PEAR-Archiv

Exkurs lokales PEAR

- `$ pear config-create /foo /foo/pear.conf`

Buildschritt: integration-test

- `@covers integrationtest` Annotation
- Ausführen der Integrationstests
  - In der Dev-Umgebung
  - Gegen die Quellen im Zip-Archiv
  - Optional gegen das Phar-Archiv
  - Optional gegen das PEAR-Archiv

Exkurs lokales PEAR

- `$ pear config-create /foo /foo/pear.conf`
- `$ pear -c /foo/pear.conf config-set auto_discover 1`

Buildschritt: integration-test

- `@covers integrationtest` Annotation
- Ausführen der Integrationstests
  - In der Dev-Umgebung
  - Gegen die Quellen im Zip-Archiv
  - Optional gegen das Phar-Archiv
  - Optional gegen das PEAR-Archiv

Exkurs lokales PEAR

- `$ pear config-create /foo /foo/pear.conf`
- `$ pear -c /foo/pear.conf config-set auto_discover 1`
- `$ pear -c /foo/pear.conf install --alldeps project.tgz`

Buildschritt: verify

- Erzeugung zeitaufwändiger Qualitätsreports

Buildschritt: verify

- Erzeugung zeitaufwändiger Qualitätsreports
  - Dynamische Analyse

Buildschritt: verify

- Erzeugung zeitaufwändiger Qualitätsreports
  - Dynamische Analyse
    - Code-Coverage

Buildschritt: verify

- Erzeugung zeitaufwändiger Qualitätsreports
  - Dynamische Analyse
    - Code-Coverage
    - Optional Profiling/Performance Analyse

Buildschritt: verify

- Erzeugung zeitaufwändiger Qualitätsreports
  - Dynamische Analyse
    - Code-Coverage
    - Optional Profiling/Performance Analyse
  - Statische Analyse

Buildschritt: verify

- Erzeugung zeitaufwändiger Qualitätsreports
  - Dynamische Analyse
    - Code-Coverage
    - Optional Profiling/Performance Analyse
  - Statische Analyse
    - Copy&Paste (PHPCPD)

Buildschritt: verify

- Erzeugung zeitaufwändiger Qualitätsreports
  - Dynamische Analyse
    - Code-Coverage
    - Optional Profiling/Performance Analyse
  - Statische Analyse
    - Copy&Paste (PHPCPD)
    - Coding Standard (CodeSniffer)

Buildschritt: verify

- Erzeugung zeitaufwändiger Qualitätsreports
  - Dynamische Analyse
    - Code-Coverage
    - Optional Profiling/Performance Analyse
  - Statische Analyse
    - Copy&Paste (PHPCPD)
    - Coding Standard (CodeSniffer)
    - Metriken (PHP_Depend)

Buildschritt: verify

- Erzeugung zeitaufwändiger Qualitätsreports
  - Dynamische Analyse
    - Code-Coverage
    - Optional Profiling/Performance Analyse
  - Statische Analyse
    - Copy&Paste (PHPCPD)
    - Coding Standard (CodeSniffer)
    - Metriken (PHP_Depend)
    - Code Smells, Unused Code usw. (PHPMD)

Buildschritt: install

- Lokale Installation

Buildschritt: install

- Lokale Installation
  - Applikation

Buildschritt: install

- Lokale Installation
  - Applikation
    - Konfiguriertes Verzeichnis

Buildschritt: install

- Lokale Installation
  - Applikation
    - Konfiguriertes Verzeichnis
  - PEAR-Archiv

Buildschritt: install

- Lokale Installation
  - Applikation
    - Konfiguriertes Verzeichnis
  - PEAR-Archiv
    - Lokale PEAR-Installation

Buildschritt: install

- Lokale Installation
  - Applikation
    - Konfiguriertes Verzeichnis
  - PEAR-Archiv
    - Lokale PEAR-Installation
    - Spezifiziertes Environment (Default Snapshot)

Buildschritt: install

- Lokale Installation
  - Applikation
    - Konfiguriertes Verzeichnis
  - PEAR-Archiv
    - Lokale PEAR-Installation
    - Spezifiziertes Environment (Default Snapshot)
  - Phar-Archiv

Buildschritt: install

- Lokale Installation
  - Applikation
    - Konfiguriertes Verzeichnis
  - PEAR-Archiv
    - Lokale PEAR-Installation
    - Spezifiziertes Environment (Default Snapshot)
  - Phar-Archiv
    - **TODO:** Apache-Ivy Repository

Buildschritt: deploy

- Remote Installation

Buildschritt: deploy

- Remote Installation
  - Applikation

Buildschritt: deploy

- Remote Installation
  - Applikation
    - Konfigurierter Server + Verzeichnis

Buildschritt: deploy

- Remote Installation
  - Applikation
    - Konfigurierter Server + Verzeichnis
    - Spezifiziertes Environment (Testing, Staging usw.)

Buildschritt: deploy

- Remote Installation
  - Applikation
    - Konfigurierter Server + Verzeichnis
    - Spezifiziertes Environment (Testing, Staging usw.)
    - Optional Datenbank-Migration mit DbDeploy

Buildschritt: deploy

- Remote Installation
  - Applikation
    - Konfigurierter Server + Verzeichnis
    - Spezifiziertes Environment (Testing, Staging usw.)
    - Optional Datenbank-Migration mit DbDeploy
  - PEAR-Archiv

Buildschritt: deploy

- Remote Installation
  - Applikation
    - Konfigurierter Server + Verzeichnis
    - Spezifiziertes Environment (Testing, Staging usw.)
    - Optional Datenbank-Migration mit DbDeploy
  - PEAR-Archiv
    - Remote Pirum-Server

Buildschritt: deploy

- Remote Installation
  - Applikation
    - Konfigurierter Server + Verzeichnis
    - Spezifiziertes Environment (Testing, Staging usw.)
    - Optional Datenbank-Migration mit DbDeploy
  - PEAR-Archiv
    - Remote Pirum-Server
    - Spezifiziertes Environment (Default Snapshot)

Buildschritt: deploy

- Remote Installation
  - Applikation
    - Konfigurierter Server + Verzeichnis
    - Spezifiziertes Environment (Testing, Staging usw.)
    - Optional Datenbank-Migration mit DbDeploy
  - PEAR-Archiv
    - Remote Pirum-Server
    - Spezifiziertes Environment (Default Snapshot)
  - Phar-Archiv

Buildschritt: deploy

- Remote Installation
  - Applikation
    - Konfigurierter Server + Verzeichnis
    - Spezifiziertes Environment (Testing, Staging usw.)
    - Optional Datenbank-Migration mit DbDeploy
  - PEAR-Archiv
    - Remote Pirum-Server
    - Spezifiziertes Environment (Default Snapshot)
  - Phar-Archiv
    - **TODO:** Apache-Ivy Remote Repository

Erweiterbarkeit I

- Die Abfolge der Buildschritte ist fest vorgegeben

Erweiterbarkeit I

- Die Abfolge der Buildschritte ist fest vorgegeben
  - Jeder Buildschritt wird aber erst durch Erweiterungen mit Leben befüllt

Erweiterbarkeit I

- Die Abfolge der Buildschritte ist fest vorgegeben
  - Jeder Buildschritt wird aber erst durch Erweiterungen mit Leben befüllt
    - Je Buildschritt existiert ein Pre-Hook

Erweiterbarkeit I

- Die Abfolge der Buildschritte ist fest vorgegeben
  - Jeder Buildschritt wird aber erst durch Erweiterungen mit Leben befüllt
    - Je Buildschritt existiert ein Pre-Hook
    - Und ein passender Post-Hook

Erweiterbarkeit I

- Die Abfolge der Buildschritte ist fest vorgegeben
  - Jeder Buildschritt wird aber erst durch Erweiterungen mit Leben befüllt
    - Je Buildschritt existiert ein Pre-Hook
    - Und ein passender Post-Hook
    - Hierdurch können projektspezifische Tasks in den Build integriert werden.

Erweiterbarkeit I

- Die Abfolge der Buildschritte ist fest vorgegeben
  - Jeder Buildschritt wird aber erst durch Erweiterungen mit Leben befüllt
    - Je Buildschritt existiert ein Pre-Hook
    - Und ein passender Post-Hook
    - Hierdurch können projektspezifische Tasks in den Build integriert werden.
  - Eine Reihe von Tasks sind von Haus aus bereits integriert

Erweiterbarkeit I

- Die Abfolge der Buildschritte ist fest vorgegeben
  - Jeder Buildschritt wird aber erst durch Erweiterungen mit Leben befüllt
    - Je Buildschritt existiert ein Pre-Hook
    - Und ein passender Post-Hook
    - Hierdurch können projektspezifische Tasks in den Build integriert werden.
  - Eine Reihe von Tasks sind von Haus aus bereits integriert
    - PHPUnit

Erweiterbarkeit I

- Die Abfolge der Buildschritte ist fest vorgegeben
  - Jeder Buildschritt wird aber erst durch Erweiterungen mit Leben befüllt
    - Je Buildschritt existiert ein Pre-Hook
    - Und ein passender Post-Hook
    - Hierdurch können projektspezifische Tasks in den Build integriert werden.
  - Eine Reihe von Tasks sind von Haus aus bereits integriert
    - PHPUnit
    - PHPMD, PHPCPD, CodeSniffer, ...

Erweiterbarkeit I

- Die Abfolge der Buildschritte ist fest vorgegeben
  - Jeder Buildschritt wird aber erst durch Erweiterungen mit Leben befüllt
    - Je Buildschritt existiert ein Pre-Hook
    - Und ein passender Post-Hook
    - Hierdurch können projektspezifische Tasks in den Build integriert werden.
  - Eine Reihe von Tasks sind von Haus aus bereits integriert
    - PHPUnit
    - PHPMD, PHPCPD, CodeSniffer, ...
    - PEAR- & Phar-Packaging

Erweiterbarkeit I

- Die Abfolge der Buildschritte ist fest vorgegeben
  - Jeder Buildschritt wird aber erst durch Erweiterungen mit Leben befüllt
    - Je Buildschritt existiert ein Pre-Hook
    - Und ein passender Post-Hook
    - Hierdurch können projektspezifische Tasks in den Build integriert werden.
  - Eine Reihe von Tasks sind von Haus aus bereits integriert
    - PHPUnit
    - PHPMD, PHPCPD, CodeSniffer, ...
    - PEAR- & Phar-Packaging
    - Pirum

Erweiterbarkeit II

- Seit Version 1.8 gibt es in Apache-Ant die so genannten Extension-Points. Ein praktischer Mechanismus für modulare Buildskripte

```
<project name="Build-Commons-Deploy">
  <target name="deploy"
          depends="-deploy:before~hook,
                   -deploy:deploy,
                   -deploy:after~hook" />

  <extension-point name="-deploy:before~hook" />

  <extension-point name="-deploy:after~hook" />

  <target name="-deploy:deploy"> ... </target>
</project>
```

Erweiterbarkeit III

```
1   <project name="My-Deploy-Extension">
2
3     <target name="-my:deploy:before~hooked"
4             extensionOf="-deploy:before~hook">
5       <!-- Stuff to be done before we deploy -->
6     </target>
7
8     <target name="-my:deploy:after~hooked"
9             extensionOf="-deploy:after~hook">
10      <!-- Stuff done after we deployed -->
11    </target>
12
13  </project>
```

Outline

Motivation

Die Idee

Anforderungen

Features

Roadmap

Fazit

Tools auf der TODO-Liste

- JSLint

Tools auf der TODO-Liste

- JSLint
- Selenium

Tools auf der TODO-Liste

- JSLint
- Selenium
- Canoo Webtest

Tools auf der TODO-Liste

- JSLint
- Selenium
- Canoo Webtest
- JMeter

Tools auf der TODO-Liste

- JSLint
- Selenium
- Canoo Webtest
- JMeter
- Apache-Ivy

Tools auf der TODO-Liste

- JSLint
- Selenium
- Canoo Webtest
- JMeter
- Apache-Ivy
- Infrastruktur

Tools auf der TODO-Liste

- JSLint
- Selenium
- Canoo Webtest
- JMeter
- Apache-Ivy
- Infrastruktur
  - VirtualBox

Tools auf der TODO-Liste

- JSLint
- Selenium
- Canoo Webtest
- JMeter
- Apache-Ivy
- Infrastruktur
  - VirtualBox
  - VMWare

Tools auf der TODO-Liste

- JSLint
- Selenium
- Canoo Webtest
- JMeter
- Apache-Ivy
- Infrastruktur
  - VirtualBox
  - VMWare
  - Puppet

Tools auf der TODO-Liste

- JSLint
- Selenium
- Canoo Webtest
- JMeter
- Apache-Ivy
- Infrastruktur
  - VirtualBox
  - VMWare
  - Puppet
  - ...

Outline

Motivation

Die Idee

Anforderungen

Features

Roadmap

Fazit

Fazit

- Einsatz in Open-Source-Projekten

Fazit

- Einsatz in Open-Source-Projekten
  - Messbare Zeitersparnis beim Projektwechsel

Fazit

- Einsatz in Open-Source-Projekten
  - Messbare Zeitersparnis beim Projektwechsel
  - Einheitlicher Prozess auf allen Systemen

Fazit

- Einsatz in Open-Source-Projekten
  - Messbare Zeitersparnis beim Projektwechsel
  - Einheitlicher Prozess auf allen Systemen
    - Development

Fazit

- Einsatz in Open-Source-Projekten
  - Messbare Zeitersparnis beim Projektwechsel
  - Einheitlicher Prozess auf allen Systemen
    - Development
    - Continuous-Integration

Fazit

- Einsatz in Open-Source-Projekten
  - Messbare Zeitersparnis beim Projektwechsel
  - Einheitlicher Prozess auf allen Systemen
    - Development
    - Continuous-Integration
    - Test/Stage

Fazit

- Einsatz in Open-Source-Projekten
  - Messbare Zeitersparnis beim Projektwechsel
  - Einheitlicher Prozess auf allen Systemen
    - Development
    - Continuous-Integration
    - Test/Stage

Bei extrem geringem Konfigurationsaufwand

Fazit

- Einsatz in Open-Source-Projekten
  - Messbare Zeitersparnis beim Projektwechsel
  - Einheitlicher Prozess auf allen Systemen
    - Development
    - Continuous-Integration
    - Test/Stage

Bei extrem geringem Konfigurationsaufwand

- Einsatz in Kundenprojekten

Fazit

- Einsatz in Open-Source-Projekten
  - Messbare Zeitersparnis beim Projektwechsel
  - Einheitlicher Prozess auf allen Systemen
    - Development
    - Continuous-Integration
    - Test/Stage

Bei extrem geringem Konfigurationsaufwand

- Einsatz in Kundenprojekten
  - Der Buildprozess ließ sich recht leicht adaptieren

Fazit

- Einsatz in Open-Source-Projekten
  - Messbare Zeitersparnis beim Projektwechsel
  - Einheitlicher Prozess auf allen Systemen
    - Development
    - Continuous-Integration
    - Test/Stage

Bei extrem geringem Konfigurationsaufwand

- Einsatz in Kundenprojekten
  - Der Buildprozess ließ sich recht leicht adaptieren
  - Nicht realisierbare Anforderungen zeigten Schwachstellen...

Fazit

- Einsatz in Open-Source-Projekten
  - Messbare Zeitersparnis beim Projektwechsel
  - Einheitlicher Prozess auf allen Systemen
    - Development
    - Continuous-Integration
    - Test/Stage

Bei extrem geringem Konfigurationsaufwand

- Einsatz in Kundenprojekten
  - Der Buildprozess ließ sich recht leicht adaptieren
  - Nicht realisierbare Anforderungen zeigten Schwachstellen...
  - ...und halfen das Projekt zu verbessern

Vielen Dank für Ihre Aufmerksamkeit

Fragen? Kommentare? Kritik? Ideen?

Vielen Dank für Ihre Aufmerksamkeit

Please rate this talk at
http://joind.in/3914

Die Folien finden Sie bald unter
http://talks.qafoo.com

Kontakt

- Manuel Pichler
- manuel@qafoo.com
- @manuelp / @qafoo