

# Build-Management mit tausend Füßen

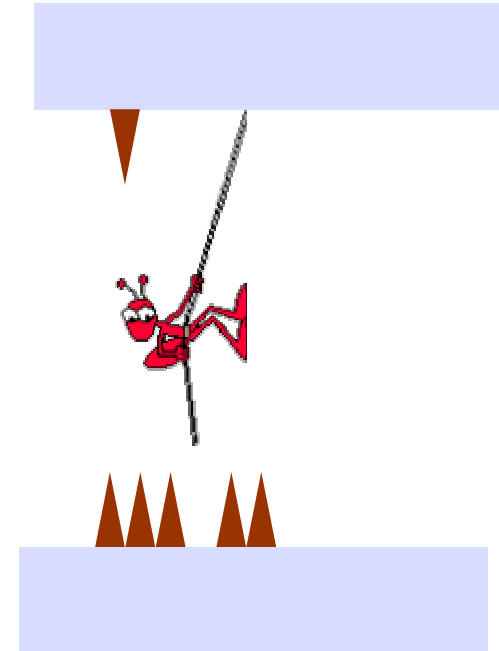
Michael Kloss ([mk@objektpark.de](mailto:mk@objektpark.de))

---

# Über meine Person ...

---

- Michael Kloss
  - Angestellt bei Peter Roßbach
  - Coach und Entwickler
  - Themen
    - J2EE-Umfeld
    - Build-Funktionalitäten
    - Test
  - Infos
    - Coming soon: Tomcat Buch
      - Tomcat <http://tomcat.objektpark.org/>
    - E-Mail : [mk@objektpark.de](mailto:mk@objektpark.de)



# Überblick

---

- Was ist Build-Management
- Centipede
- Cents & Co.
- Stand der Dinge
- Fazit

# Ziel

---

- Verständnis der Anforderungen an Build-Management
- Einsatz von Werkzeugen im Bereich Build-Management
- Möglichkeiten des Werkzeugs Centipede
- Potenziale



---

# Build-Management

# Motivation

---

- Projekterzeugnis (6-10 Entwickler)
  - 1,5 Jahre
  - ca. 600 Klassen und Tests
  - diverse Ressourcen
  - 20 benutzte Bibliotheken
    - Fremdbibliotheken
    - eigene Entwicklungen
  - CVS als Versionskontrollsystem
  - Ant als Build-Tool
- Probleme
  - Bibliotheken wurden vergessen zu verwalten
  - Dokumentation wurde nicht zum Stand der Implementierung gepflegt
    - keine Verknüpfung mit den Releases
  - Unterschiedliche Abhängigkeiten der Versionen
    - Bibliotheken passten nicht zum Versionstand

# Warum Build-Management

---

- komplexere Projekte
- größere Teams mit entsprechenden Subprojekten
- verteiltes Arbeiten
- viele Abhängigkeiten
  - Fremdbibliotheken
  - internen Projekterzeugnissen
  - Subprojekten

# Warum Build-Management

---

- Historische Konsequenz
- Erzeugnisse für die Ewigkeit...
  - Keiner weiß, wie lange heutige Entwicklungen eingesetzt werden
- Wie kann ein Entwickler in 20 Jahren heutige Entwicklungen nutzen?
  - Alle Abhängigkeiten müssen verfügbar sein
  - Eine Dokumentation aller Komponenten sollte existieren
  - Verwaltung von Erzeugnissen

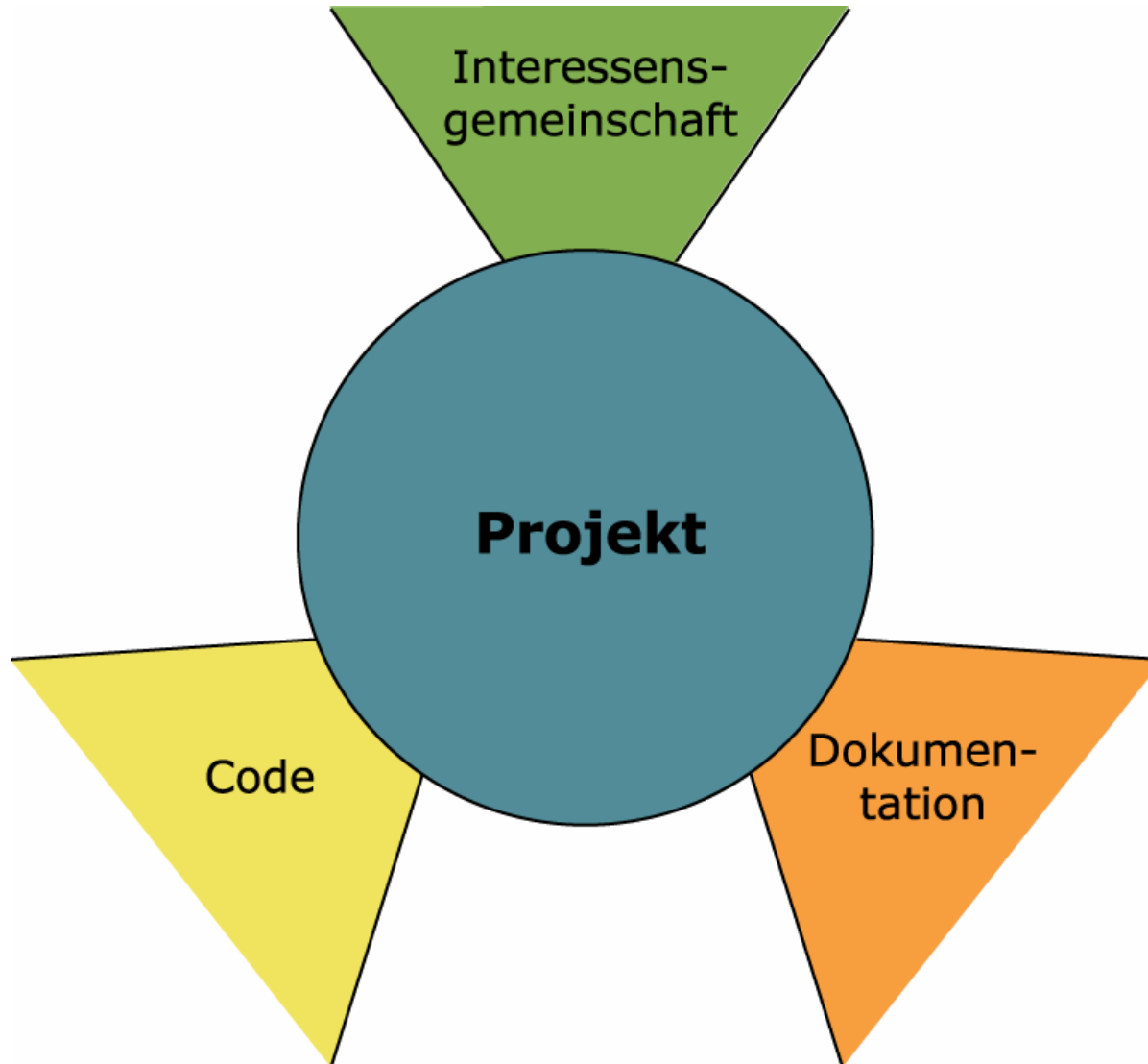
## Was gehört dazu ...

---

- Organisation des Projekts
  - Struktur
  - externe Abhängigkeiten
- Modularisierung des Projekts
  - Definition von Subprojekten
  - interne Abhängigkeiten
- Automatisierungsprozesse
- Konfigurationsmanagement
- Werkzeuge
- Testbarkeit

# Community Project Documentation Code (CoPDoC)

---



# Bedeutung des Konfigurationsmanagements

---

- Beispiel
  - Entwickler genießen den Komfort Versionen zu verwalten
  - Die Möglichkeit zwischen Versionen zu springen
  
- Gibt es denn dann auch eine Verwaltung für Erzeugnisse, Dokumentation und Werkzeuge?
  
- Definition seiner Abhängigkeiten zu anderen Erzeugnissen
- Eigene Repositories für andere Erzeugnisse
  
- Definition seiner Abhängigkeiten zu anderen Repositories...

# Einige bekannte Werkzeuge

---

- Build
  - Ant
  - Make
- Konfigurationsmanagement
  - CVS
  - Subversion
  - PVCS
  - ClearCase
  - VSS
  - Perforce
- OpenSource Build-Management-Tools
  - Maven
  - ... Centipede



---

# Centipede

# Mein Sandkasten

- Der Pizzashop
  - Struts als MVC
  - Tiles als Layout-Manager
  - Castor für XML-Serialisierung
  - Object Relational Bridge (OBJ)
  - Commons-Logging
  - Log4J
  
- => große Abhängigkeit zu Fremdbibliotheken



[Anmelden](#)  
[Kontakt](#)

Willkommen beim Pizza Service Online.



Sind Sie noch nicht registriert? Klicken [hier](#) um sich zu registrieren.

webdev Tutorial by [Peter Roßbach](#) & [Michael Kloss](#) © 2003

# Pizzashop – Build-Skript

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE project [
    <ENTITY common SYSTEM "file:./common.xml">
]>

<project name="pizzashop" default="cleancompile" basedir=". ">

    <target name="jalopy" description="Formats Code">
        <jalopy fileformat="auto"
            convention="${basedir}/jalopy.xml"
            history="file"
            historymethod="adler32"
            loglevel="info"
            threads="2"
            classpathref="compile.classpath">
            <fileset dir="src">
                <include name="**/*.java" />
            </fileset>
        </jalopy>
    </target>

    <target name="checkstyle" depends="jalopy"
        description="Generates a report of code convention violations.">
        <mkdir dir="report"/>
        <checkstyle config="${basedir}/checkstyle.xml"
            failureProperty="checkstyle.failure"
            failOnViolation="false">
            <formatter type="xml" tofile="report/checkstyle_report.xml"/>
            <fileset dir="src" includes="**/*.java"/>
        </checkstyle>

        <style in="report/checkstyle_report.xml" out="report/checkstyle_report.html" style="xsl/checkstyle-noframes-sorted.xsl"/>
        <copy todir="./../cruiserweb/reports/${cctimestamp}/checkstyle">
            <fileset dir="report">
                <exclude name="**/*.xml"/>
                <exclude name="**/*.svg"/>
                <include name="**/checkstyle*.html"/>
            </fileset>
        </copy>
    </target>

    <target name="jdepend">
        <delete file="${basedir}/report/jdepend-report.xml"/>
        <jdepend format="xml" outfile="${basedir}/report/jdepend-report.xml">
            <sourcepath>
                <pathelement location="build/WEB-INF/classes" />
            </sourcepath>
            <classpath refid="compile.classpath" />
        </jdepend>

        <style basedir="${basedir}/report" in="report/jdepend-report.xml" out="report/graph.svg"
            style="${basedir}/xsl/graph.xsl" />

        <style basedir="report" destid="report"
            includes="jdepend-report.xml"
            style="${basedir}/xsl/jdepend.xsl" />
        <copy todir="./../cruiserweb/reports/${cctimestamp}/jdepend">
            <fileset dir="report">
                <exclude name="**/*.xml"/>
                <include name="**/jdepend*.html"/>
                <include name="**/*.svg"/>
            </fileset>
        </copy>
    </target>

    <target name="compile" depends="prepare, validate.xml.files, checkstyle" description="Compile Java sources">
        <echo message="${catalina.home}"/>
        <!-- Compile Java classes as necessary -->
        <mkdir dir="${build.home}/WEB-INF/classes">
        <javac srcdir="src" destdir="${build.home}/WEB-INF/classes" debug="${compile.debug}" deprecation="${compile.deprecation}"
            optimize="${compile.optimize}">
            <classpath refid="compile.classpath"/>
        </javac>
        <!-- Copy associated resource files -->
        <copy todir="${build.home}/WEB-INF/classes">
            <fileset dir="src" includes="**/*.properties"/>
        </copy>
    </target>

    <target name="jspc" description="validate jsp with catalina Jasper JSPC">
        <ant antfile="jspc.xml" target="jspc" />
    </target>

    <target name="validate.xml.files" description="Validates all xml-files in project">
        <xmlvalidate classpathref="libs.classpath">
            <fileset refid="validation.xml.list"/>
            <xmlcatalog refid="commons.dtd"/>
        </xmlvalidate>
        <antcall target="validate.validator"/>
    <!-- <antcall target="validate.wellformedness"/> -->
    </target>

    <target name="validate.validator" description="Validates validator-rules.xml">
        <echo message="Must be outsourced because uses same PublicId as validator.xml but different DTD"/>
        <xmlvalidate classpathref="libs.classpath">
            <fileset dir="${basedir}">
                <include name="**/validator-rules.xml"/>
            </fileset>
            <dtd publicid="://Apache Software Foundation/DTD Commons Validator Rules Configuration 1.0/EN" location="${dtd.dir}/validator-
                rules_1_1.dtd"/>
        </xmlvalidate>
    </target>

    <target name="validate.wellformedness" description="Test XML-Files without DTD if Wellformed">
        <echo message="${basedir}" />
        <xmlvalidate classpathref="libs.classpath" lenient="true">
            <fileset refid="wellformed.xml.list"/>
        </xmlvalidate>
    </target>

    <target name="javadoc" depends="compile" description="Create Javadoc API documentation">
        <mkdir dir="docs/api"/>
        <javadoc sourcepath="src" destdir="docs/api" packagenames="org.objektpark.pizzashop.*"/>
    </target>

    <target name="clean" description="Delete old build and dist directories">
        <delete dir="${build.home}"/>
        <delete dir="${build.jspc}"/>
        <delete dir="${webapps.home}"/>
    </target>

    <target name="cleancompile" depends="clean, compile, jspc, jdepend" description="Clean build and dist, then compile">
    </target>

    <target name="start" depends="compile, jspc" description="start test tomcat">
        <ant antfile="tomadmin.xml" target="tomstart"/>
    </target>

    <target name="stop" description="stop test tomcat">
        <ant antfile="tomadmin.xml" target="tomstop"/>
    </target>

</project>
```

# Pizzashop – Build-Skript mit Centipede

---

```
<?xml version="1.0"?>
<project default="code" basedir="." name="project build file">
<description>Pizzashop Project</description>

<taskdef resource="centipede"/>
<centipede/>

<importcent name="antidote"/>
<importcent name="makecent"/>
<importcent name="jalopy"/>
<importcent name="java"/>
<importcent name="junit"/>
<importcent name="forrest"/>
<importcent name="checkstyle"/>
<importcent name="gump"/>
<importcent name="javasrc"/>
<importcent name="jdiff"/>
<importcent name="umldoclet"/>

    <target name="code" depends="compile,test,package"/>
    <target name="metrics" depends="checkstyle, report"/>
    <target name="docs" depends="javasrc, javadocs, site"/>
    <target name="gump" description="Target used by Gump"/>
</project>
```

# Was genau ist denn jetzt Centipede?

---

- Ant-Aufsatz
- Definiert Templates für Ant
  - und kann diese importieren
- Strukturiert das Projekt durch deklarative Beschreibungen
- Verwaltet Abhängigkeiten zu
  - Bibliotheken
  - anderen Projekten

# Was genau ist denn jetzt Centipede?

---

- Die Verbindung bestehender Werkzeuge
  - Apache Ant
  - Krysalis Ruper
  - Apache Forrest
  - Apache Gump
  
- Zusätzlich
  - Centipede-Task für Ant
  - kleinere Ant-Task-Erweiterungen
  - Bereitstellung sog. *Cents*

# Build-Umgebung

---

- Ant als Build-Werkzeug
  - <http://ant.apache.org>
- deFacto-Standard im Bereich Java
  - inzwischen auch in anderen Bereichen
- basiert auf Java
- hochportabel
- deklarativ
- viele Erweiterungen

# Bibliotheken-Verwaltung

---

- Krysalis Ruper
  - <http://www.krysalis.org/ruper>
  - Zuständig für
    - Jar-Abhängigkeiten
    - Projekt-Abhängigkeiten
  - Remote-Zugriff auf entfernte Repositories per HTTP
  - Anlegen eines lokalen Repositories
    - Jar-Repository
    - Cent-Repository
  - Namenskonventionen nach Jakarta JAR Archive Repository (*jjar*)
    - **jakarta-commons-sandbox**

# Dokumentation

---

- Apache Forrest
  - <http://xml.apache.org/forrest>
  - Erstellung der Projekt-Dokumentation
- XML Standard-orientiertes Dokumentations-Framework
  - unterstützt diverse Format
  - basiert auf Cocoon
  - XSLT-Stylesheets
  - Schemas
  - benutzt Skins für die Darstellung
- Vervollständigung von Gump
- Priorität auf "low startup cost"

# Abhängigkeiten & Kontinuierliche Integration

---

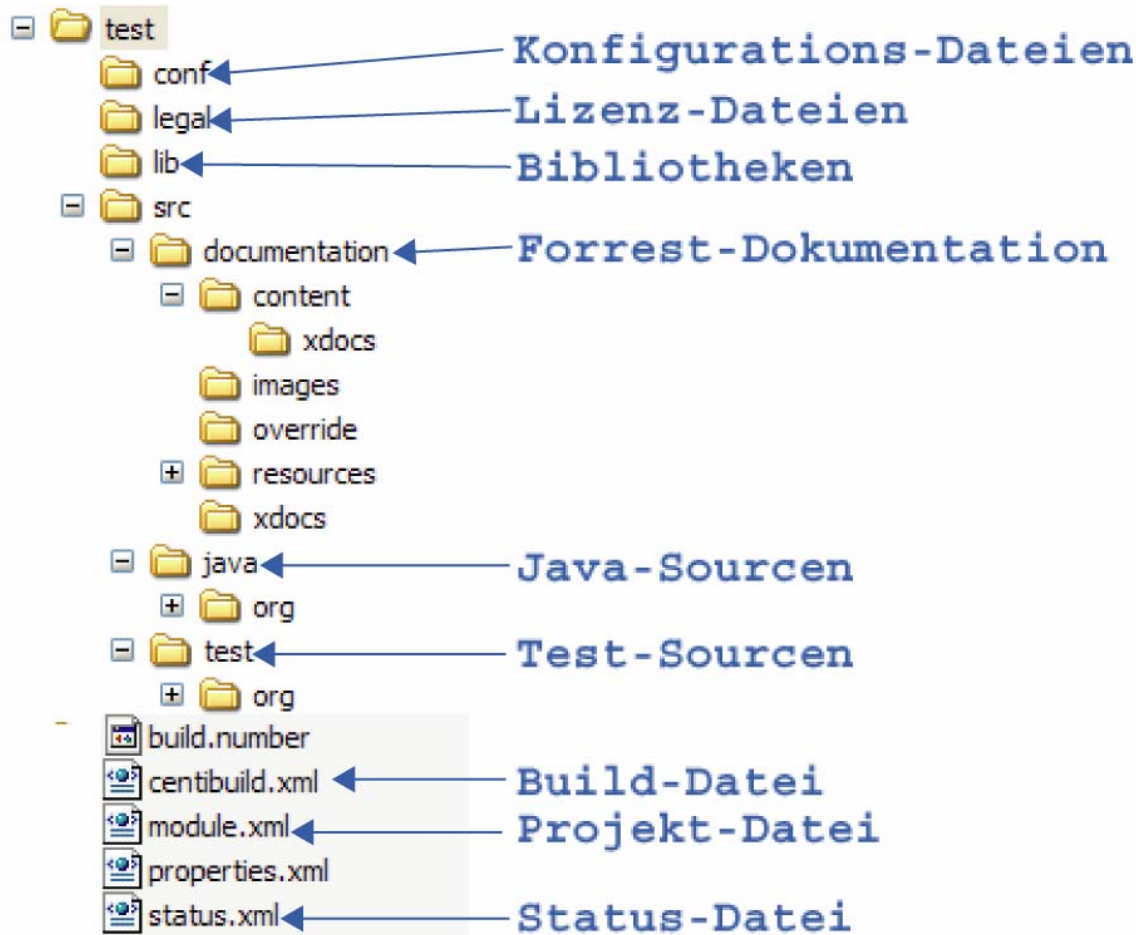
- Apache Gump
  - <http://jakarta.apache.org/gump>
  - Kontinuierliche Integration
- Build-Status von Projekte
- Abhängigkeit von Projekten
- Kontinuierliche Integration

# Abhängigkeiten & Kontinuierliche Integration

<ul style="list-style-type: none"> <li><a href="#">commons-cli</a></li> <li><a href="#">commons-codec</a></li> </ul>	05:45:06	<a href="#">jakarta-bsf</a>	<b>FAILED</b>
<ul style="list-style-type: none"> <li><a href="#">commons-jelly-tags-impl</a></li> <li><a href="#">commons-jelly-tags-html</a></li> <li><a href="#">commons-jelly-tags-http</a></li> <li><a href="#">commons-jelly-tags-interaction</a></li> <li><a href="#">commons-jelly-tags-jetty</a></li> <li><a href="#">commons-jelly-tags-jms</a></li> <li><a href="#">commons-jelly-tags-jsl</a></li> <li><a href="#">commons-jelly-tags-junit</a></li> <li><a href="#">commons-jelly-tags-log</a></li> <li><a href="#">commons-jelly-tags-obj</a></li> </ul>	05:55:10	<a href="#">avalon-fortress-tools</a>	SUCCESS
	05:55:20	<a href="#">excalibur-datasource</a>	SUCCESS
	05:55:27	<a href="#">avalon-components</a>	SUCCESS
	05:55:34	<a href="#">beanshell</a>	PREREQ FAILURE - jakarta-bsf
	05:55:34	<a href="#">sommers-net</a>	SUCCESS
	05:56:43	<a href="#">excalibur-extension</a>	SUCCESS
	05:56:55	<a href="#">excalibur-meta-sourceresolve</a>	SUCCESS
<ul style="list-style-type: none"> <li><a href="#">dist-bsf</a></li> <li><a href="#">dist-xalan2</a></li> <li><a href="#">dist-xerces</a></li> <li><a href="#">dist-xerces1</a></li> <li><a href="#">dom4j</a></li> <li><a href="#">excalibur-compatibility</a></li> <li><a href="#">excalibur-component</a></li> <li><a href="#">excalibur-configuration</a></li> <li><a href="#">excalibur-datasource</a></li> <li><a href="#">excalibur-event</a></li> <li><a href="#">excalibur-extension</a></li> <li><a href="#">excalibur-i18n</a></li> <li><a href="#">excalibur-instrument</a></li> <li><a href="#">excalibur-instrument-client</a></li> <li><a href="#">excalibur-instrument-manager</a></li> <li><a href="#">excalibur-instrument-manager-altmi</a></li> <li><a href="#">excalibur-lifecycle</a></li> <li><a href="#">excalibur-logger</a></li> <li><a href="#">excalibur-meta-sourceresolve</a></li> <li><a href="#">excalibur-monitor</a></li> <li><a href="#">excalibur-pool</a></li> <li><a href="#">excalibur-sourceresolve</a></li> <li><a href="#">excalibur-store</a></li> </ul>	06:06:39	<a href="#">jakarta-cactus-sample-servlet-12</a>	SUCCESS
	06:07:50	<a href="#">javagroups</a>	PREREQ FAILURE - beanshell
	06:07:51	<a href="#">jfor</a>	SUCCESS
	06:07:58	<a href="#">jrefactory-pretty</a>	SUCCESS
	06:08:14	<a href="#">xdoclet-compile-core</a>	SUCCESS
	06:08:22	<a href="#">xdoclet-xdoclet-module-prepare</a>	SUCCESS
	06:08:29	<a href="#">spice-configkit</a>	SUCCESS
	06:08:41	<a href="#">spice-salt-pe</a>	SUCCESS
	06:08:53	<a href="#">tomcat-catalina</a>	SUCCESS
	06:09:09	<a href="#">jakarta-tomcat-coyote</a>	SUCCESS
	06:09:23	<a href="#">jakarta-tomcat-catalina</a>	PREREQ FAILURE - javagroups
	06:09:23	<a href="#">xml-fop-maintenance</a>	<b>FAILED</b>
	06:10:03	<a href="#">ant-embed-optional</a>	SUCCESS
	06:10:07	<a href="#">bootstrap-obj</a>	SUCCESS
	06:12:27	<a href="#">commons-attributes</a>	SUCCESS
	06:12:34	<a href="#">commons-daemon</a>	SUCCESS
	06:12:41	<a href="#">commons-http</a>	SUCCESS

# Centipede Aufbau

## □ Default-Template



# Centipede Aufbau

---

- centibuild.xml
  - Build-Skript für das Projekt
  - Initialisierung von Centipede
  - Import der Cents
  - Eigene Targets

# Centipede Aufbau

---

- module.xml
  - Allgemeine Beschreibung des Projekts
    - URL des Projekts
    - Remote-Verzeichnis der Seite
    - CVS-Daten
    - Definition der Repositories
    - Ziele

# Centipede Aufbau

---

- module.xml
  - Definition der Projekte
    - Abhängigkeiten zu anderen Projekten
    - Abhängigkeiten zu Bibliotheken
    - Work-Verzeichnis
    - Home-Verzeichnis
    - Code-Verzeichnis
    - Test-Verzeichnis
    - Dokumentationsverzeichnis
    - Package-Angabe
    - Versionierungsangabe

# Centipede Aufbau

## □ Beispiel

```
<module name="pizzashop-project" fullname="Pizzashop WebDev
Project">
<url href="http://pizzashop.objektpark.org"/>
<site hostname="objektpark.net"
remotedir="/home/pizzashop"/>
<cvs repository="objektpark.net"
host-prefix=""
dir="/home/repository"
module="pizzashop"
href="http://pizzashop.objektpark.org"/>

<jars repository="maven" url="http://www.ibiblio.org/maven"/>
<jars repository="micha" url="/lib/srclib"/>

<description>
Pizzashop WebDev Project
</description>

<detailed>... </detailed>

<what><goal>...</goal> </what>

<why>...</why>

<vendor>Objektpark</vendor>

<licence legal=".legal">...</licence>
```

```
<project name="pizzashop-project" fullname="Pizzashop WebDev Project">
<version major="1" minor="1"
fix ="2"
tag="HEAD"/>

<package>org.objektpark.pizzashop</package>

<depend project="log4j" inherit="all" export="true"/>
<depend project="castor" inherit="all" export="true"/>
<depend project="commons-logging" inherit="all" export="true"/>
<depend project="commons-beanutils" inherit="all" export="true"/>
<depend project="struts" inherit="all" version="1.1" export="true"/>
<depend project="servletapi" inherit="all" version="2.3" export="true"/>

<ant target="gump" vm="1.2"/>

<!-- Work dirs to be included in classpath -->
<work nested="build/classes"/>

<home nested="build"/>

<code type="java/plain" dir="src"/>
<test type="test/junit" dir="test"/>
<documentation type="xml/forrest" dir="documentation"/>

</project>

</module>
```

# Centipede Aufbau

---

- properties.xml
  - Datei für
    - eigene Property-Variablen
    - überschreiben von Cent-Properties
  - Ant-XML-Property-Format
    - ignoreRoot ist hierbei eingeschaltet
  - Property-Aufbau
    - test.code.dir
      - `<test><code><dir>/meinVerzeichnis/test</dir></code></test>`
      - `<test><code dir="/meinVerzeichnis/test"/></test>`
- status.xml
  - Informationen zum Status des Projekts
    - Entwickler
    - ToDo's
    - Änderungen

## Getting started...

---

- Pfade setzen
  - CENTIPEDE\_HOME
  - FORREST\_HOME
- Es existiert ein Template für Centipede 1.0.0\_b5
  - krysalis-project-template.tar.gz
    - Unter Windows dieses Archiv auspacken
    - Unter Unix centseed aufrufen
- Aufruf mit *cent* oder
- mit *cent* <target\_name>

# Getting started...

---

```
C:\ C:\WINDOWS\System32\cmd.exe - cent
```

```
-----  
pizzashop-project [2003]  
-----
```

```
Using Apache Ant version 1.6alpha compiled on January 1 2003  
Build file C:\alphaworks\pizzashop_centipede\centibuild.xml  
-----
```

```
These are the most common build targets.  
You can also invoke them directly; see build.xml for more info.  
Builds will be in /build directory, distributions in /dist.
```

```
all ----- creates the jars and the site  
compile ----- compiles the source code  
test ----- performs the junit tests  
jar ----- create the jar files  
docs ----- generates the html docs - clean not needed  
javadocs ----- generates the API documentation  
site ----- generates the html site (docs+reports)  
clean ----- cleans the build directory  
dist ----- creates src and bin distributions
```

```
Please select a target  
compile
```

# Migration eines Projekts

---

- Pizzashop-Migration
  - Benutzen des ursprünglichen Build-Skripts
  - Anpassen der Projektbeschreibung
  - Anpassen der Projektpfade
  - Schrittweise umstellen der vorhandenen Targets auf Cents
  - Erzeugen eines eigenen Cents für die Erstellung eines Web-Archivs



---

# Cents & Co

# Keinen Cent wert?

---

## □ Cents

- Import einfach über `<importcent>`:

```
<importcent name="java"/>
```

```
<importcent name="junit"/>
```

```
<importcent name="forrest"/>
```

```
<importcent name="checkstyle"/>
```

- Danach kann jeder Target aus den importierten Cents genutzt werden

## □ Cent = Ant-Template

- Jedes Cent hat

- Main-Targets

- Helper-Targets

- DefaultProperties

- Reines Ant

- MakeCent, als Cent für Cents

# Cents

---

## □ Beispiel Cent "junit"

### ■ Targets

#### □ failsafe-test

- Führt die Tests aus ohne mit einem Fehler abubrechen; wichtig um Reports mit den Fehlern zu erzeugen

#### □ test

- Führt die Tests aus. Durch die Property `junit.cent.extra.sysproperty` in der `properties.xml` können System-Properties hinzugefügt werden

#### □ report

- Erzeugt Test-Reports

### ■ Properties

#### □ `junit.cent.docs.dir`

- Default Value: `${project.build.dir}/docs/junit`

#### □ `junit.cent.build.dir.passdown`

- Default Value: `${junit.cent.build.dir}`

#### □ `project.test.dir`

- Default Value:  
`${jxpath:/references/module.xml/root/module/project[@name=$project.name]/test[@type='test/junit']/@dir}`

# Cents

---

- Cents erstellen
  - minimale Basis besteht aus drei Teilen
    1. Ant-Skript
    2. Cent-Dokumentieren
    3. Task-Definitionen
  - Bibliotheken
  - Property-Dateien
  - etc.

# Beispiel webapp-cent

---

## □ Ant-Skript

### ■ xbuild.xml

```
<?xml version="1.0"?><project default="war" name="webapp.cent">

  <property name="webapp.cent.content.xdocs"
            value="\${webapp.cent.work.dir}/xdocs"/>
  <property name="webapp.cent.webxml.file"
            value="\${project.src.dir}/web/WEB-INF/web.xml"/>
  <property name="webapp.cent.web.dir"
            value="\${project.src.dir}/web"/>
  <property name="webapp.cent.classes.dir"
            value="\${project.build.dir}/classes"/>

  <target name="war" description="builds war-archive of build-product">

    <war destfile="\${project.build.dir}/\${project.name}.war"
         webxml="\${webapp.cent.webxml.file}">
      <fileset dir="\${webapp.cent.web.dir}"/>
      <classes dir="\${webapp.cent.classes.dir}"/>
    </war>

  </target>
</project>
```

# Beispiel webapp-cent

---

## □ Dokumentation

### ■ centdoc.xml

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

```
<centdoc name="webapp.cent">
```

```
<description>Builds war from build. </description>
```

```
<detailed/>
```

```
<property name="webapp.cent.content.xdocs">
```

```
<description/>
```

```
<default>${webapp.cent.work.dir}/xdocs</default>
```

```
</property>
```

```
...
```

```
<target type="main" name="war">
```

```
<description>Formats the sourcecode</description>
```

```
<detailed/>
```

```
<property name="webapp.cent.web.dir">
```

```
</property>
```

```
</target>
```

```
...
```

```
</centdoc>
```

# Beispiel webapp-cent

- Zusammenstellen und Installieren
  - Aufruf makecent
  - Aufruf install-cent
  - Fertig...

The screenshot shows the website for the KRYVALIS COMMUNITY PROJECT. The main content area displays the page for 'junit.cent version 0.1.1-dev-20030603'. The page has a navigation menu on the left with links for 'Home', 'Cents', and 'Templates'. The main content area includes a search bar, a breadcrumb trail, and a table of targets.

**KRYVALIS COMMUNITY PROJECT** **CENTIPEDE** the Krysalis Community Project site

Home Cents Templates

Published: 06/03/2003 12:45:44  
Font size: [A-] [A] [A+]

krvalis.at.sf.net > krysalis > cents > cents > junit

## junit.cent version 0.1.1-dev-20030603

### Introduction

[What is this cent for?](#)

[Why use this cent?](#)

Stop the cut and paste waste.

### Cent Project Description

JUnit Cent

Cents

### Targets

There are three types of target in a cent xbuild.xml file. The most important, from a users perspective, is the *main* targets. These perform the main tasks of a cent. The next most important is the *helper* targets. These are intended to be used by other targets, although they can still be called from the command line if desired. Finally, there are the *internal* targets, these cannot be called from the command line and are only used by other targets.

The table below shows a summary of the main targets available in this cent. These can be called from the command line with `cent TARGET-NAME`. You can also view the [helper targets](#) and the [internal targets](#).

Name	Description
failsafe-test	Perform jUnit tests without failing; important to make it possible to generate a report on the failings
test	Perform jUnit tests. Add the junit.cent.extra.sysproperty elements in properties.xml to add sys properties.
report	Perform jUnit test reports

# K(l)eine Überraschung...

---

- Ein ähnliches Konzept wird in der Ant 1.6 Version umgesetzt

- Antlibs
- Sammlung von TaskDefs und TypeDefs
- Namespace-Unterstützung
- Beispiel Build-Skript

```
<project xmlns:local="localpresets">
  <typedef file="localpresets.xml" uri="localpresets"/>
  <local:shellscript>
    echo "hello world"
  </local:shellscript>
</project>
```

- Beispiel localpresets.xml

```
<antlib xmlns:antcontrib="antlib:net.sf.antcontrib">
  ...
  <presetdef name="shellscript">
    <antcontrib:shellscript shell="bash"/>
  </presetdef>
</antlib>
```



---

# Stand der Dinge

## Centipede heute

---

- Nick Chalko stellte folgende Gründe für die Stagnation fest
  - "Tried to accomplish to much"
  - "Hard to understand how to extend"
  - "To many dependencies"
  - **"Outsiders saw centipede as something other than ant. "**
- Viele Eigenschaften von Centipede wurden von Ant adaptiert
  - z.B. Antlibs in Version 1.6
- Ruper und Version sollen der Apache Group angeboten werden
- Die Ideen und Probleme, die Centipede aufgezeigt hat, sind der jetzige Benefit

# Centipede 2

---

## □ Centipede2 aufgesetzt

### ■ Ziele

- Keine Abhängigkeiten mehr
- keine großen Properties-Dateien
- Einfache Erklärungen
- Ant-User integrieren, um Centipede einfach zu erweitern
- Gump freundlicher

### ■ Keine Ziele mehr

- Version Management
- Multi project builds
- Dependency Management
- Auto-updating.

### ■ Keine Sorge:

- Diese Dinge sollen über Antlets möglich werden

# Fazit

---

- Centipede 1.0 Beta 5
  - Einsetzbar, mit vorhandener Funktionalität
  - Build-Funktionalität, Jar-Repository
  - Keine Weiterentwicklung auf dieser Basis
- Centipede 2
  - Frühes Alpha-Stadium
  - Einfacher zu integrieren, da in Ant installierbar
  - Voraussichtlich weniger Definitionen in XML-Dateien
    - Module.xml wird wahrscheinlich wegfallen
  - Verbessertes Abhängigkeits-Management



## Zum Schluss

---

"Build-Management ist mehr als nur ein Build-Script..."

# F&Q

---

□ Fragen und Antworten

□ News

■ <http://www.krysalis.org/centipede>

■ Mailing-Liste von Krysalis

□ <mailto:mk@objektpark.de>



## Centipede vs. Maven

	<b>Centipede</b>	<b>Maven</b>
<b>Build-System</b>	Ant und eigene Ant-Erweiterungen	Jelly, jeder Ant-Task
<b>Plugins</b>	Cents, auf Ant basierend	Jelly-Skripte
<b>Projektbeschreibung</b>	Erweiterung des Gump-Deskriptors	Eigenes XML-Schema (früher auch Gump-basiert)
<b>Multi-Project</b>	Gump-Cent	Reactor
<b>Dokumentation</b>	Forrest	Anakia