



invesdwin

Software Produktlinien Plattform

von Edwin Stang

Agenda

1. Einleitung
2. Software Produktlinie
3. Entwicklungsumgebung
4. Konfigurationsmanagement
5. Assets
6. Bindung der Variabilität
7. DDD und MDA
8. Fazit & Diskussion

Fragen gerne zwischendurch!

1. Einleitung

Entstehungsgeschichte

Motivation

1. Einleitung

- Hintergründe -

- Bachelor Thesis -> Bisherige Plattform und SPL?
 - Stattdessen Entscheidung zu neuen Business Components
 - Blaupause für Projekte statt SPL
- Meine Entscheidung SPL zuhause zu schaffen
 - Domäne Börseninvestments, SPL als Grundlage
 - Forschungsprojekt
 - 1-Mann-Entwicklung
 - niedrige/keine laufenden Kosten
 - OpenSource **Leverage**
 - **Wiederverwendung** an erster Stelle
 - F&E macht **Flexibilität** notwendig
 - **Entwicklungskomfort** hoch priorisiert
 - Es soll **Spaß** machen!

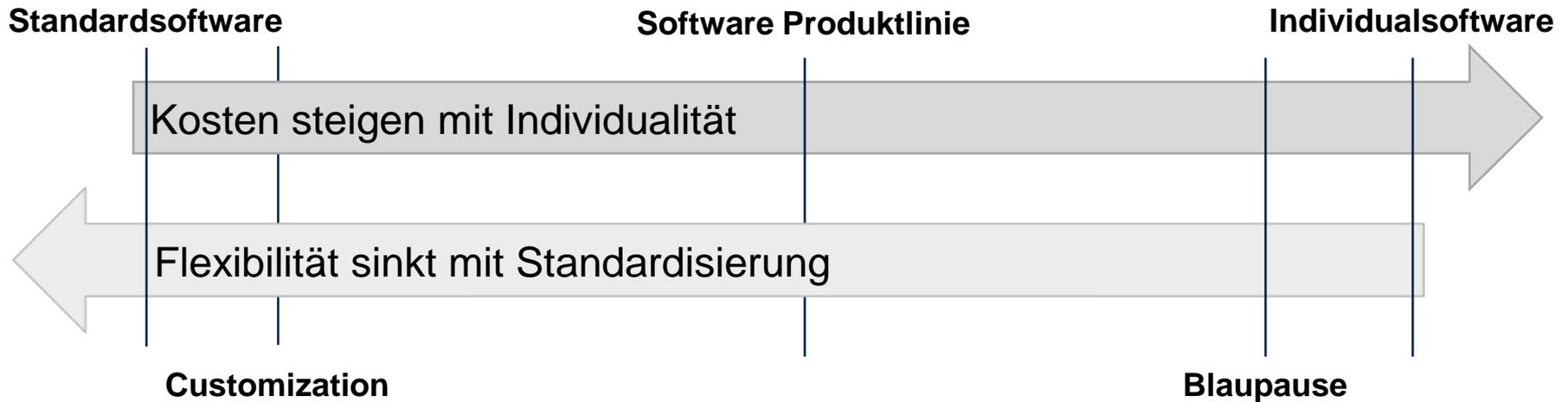


In Entwicklung seit Oktober 2009
mittlerweile recht stabil

2. Software Produktlinie

Wissenschaftlicher Hintergrund

2. Software Produktlinie - Einordnung -



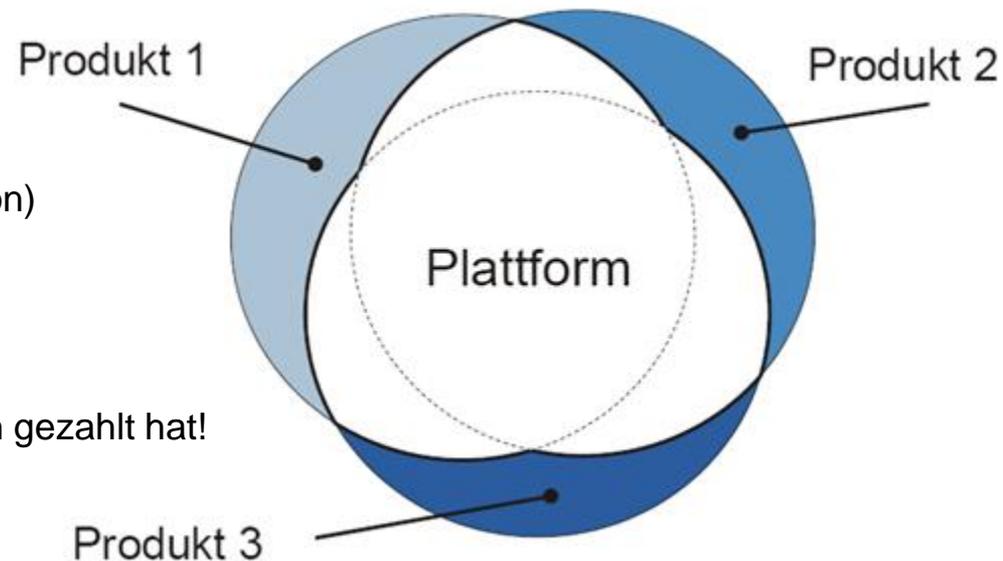
Lösung:

- ✓ Reduktion der Kosten durch **Wiederverwendung**
- ✓ Wahrung der Flexibilität durch **Variabilität**

2. Software Produktlinie

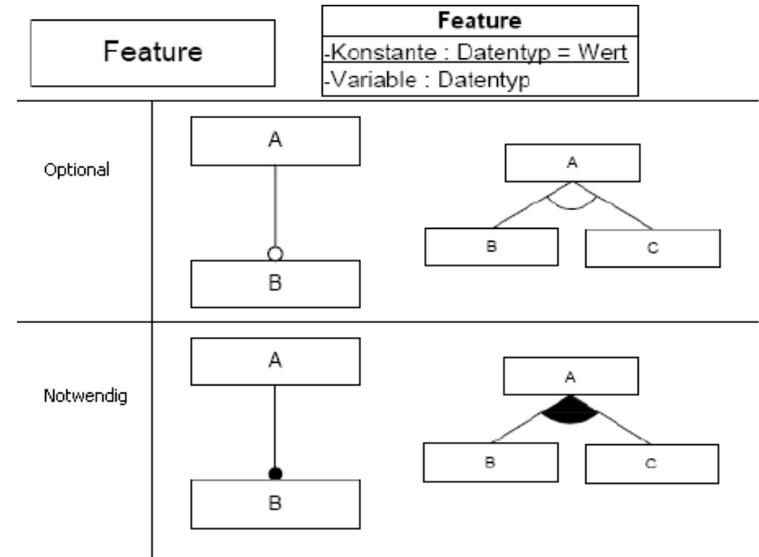
- Assets -

- Assets
 - Artefakte wie **Dokumentation, Code, Libs, Konfiguration, Produkte, Plattform**
 - SPL besteht aus - und managed Assets
- Wiederverwendung
 - Artefakte werden designed, um sie Wiederverwenden zu können
 - Schaffung von **Synergien**
 - Reduktion der Entwicklungskosten
 - Schnellere Time-To-Market
 - Copy/Paste vermeiden (Artefaktduplikation)
- Modularität
 - **Artefakte bündeln**
 - Nur das ausliefern, wofür der Kunde auch gezahlt hat!



2. Software Produktlinie - Variabilität -

- Variabilität
 - Änderbarkeit der zu nutzenden Implementation
 - Selektion von **Features**
 - Preis variabel machen und am Kunden orientieren
- Variabilitätspunkte
 - Entscheidung „wo“ Varianten gewählt werden
 - Geplante Flexibilisierungsmöglichkeit
 - Konfigurierbar oder Wählbar
 - Kunde trifft hier Entscheidungen
- Varianten
 - Features kapseln und optional wählbar machen
 - Vgl. Autobau:
 - Anhängerkupplung ja/nein
 - Sportfahrwerk vs Komfortfahrwerk
 - Kombi vs Limousine



Mercedes-Benz | Konfigurieren Sie Ihr Fahrzeug - Microsoft Inte...

Mercedes-Benz 1. Modell wählen 2. Motorisierung 3. Designlinie 4. Lack & Polster 5. Ausattung 6. Ihr Fahrzeug 7. Ihr Kontakt

5. Ausstattung

Sonderausstattung | Serienausstattung nach Preis nach Rate

Feature	Preis
<input type="checkbox"/> Sound-System Surround	765,60 EUR
<input type="checkbox"/> CD-Wechsler im Handschuhfach	491,40 EUR
<input type="checkbox"/> Radiovorrichtung (6 Lautsprecher inkl. Verstärker)	313,20 EUR
<input type="checkbox"/> Sprachbedienungs-system LINQUATRONIC	458,20 EUR
<input checked="" type="checkbox"/> COMAND APS	3.201,60 EUR
<input checked="" type="checkbox"/> Radio Audio 20 CD	765,60 EUR
<input checked="" type="checkbox"/> Radio Audio 50 APS	2.163,40 EUR
<input type="checkbox"/> Ohne	0,00 EUR
<input type="checkbox"/> Mobiltelefon-Vorrichtung mit universeller Schnittstelle	556,80 EUR

Design
Interieur
Lenkung/ Schaltung
Pakete
Räder/ Fahrwerk
Radio/ Kommunikation
Sicherheit/ Technik
Sondermodell

Ihre Wahl / Preis (EUR)

C 180 Kompressor
Limousine
Gesamtprice 36.792,28

Grundpreis 28.942,00
Designline 1.850,20
Lack/ Lack wird später festgelegt 0,00
Polster/ Polster wird später festgelegt 0,00
Sonderausstattungen Gesamt 0,00

Ihre Optionen
Fahrzeug speichern

© DaimlerChrysler, 2004-2006

Fertig

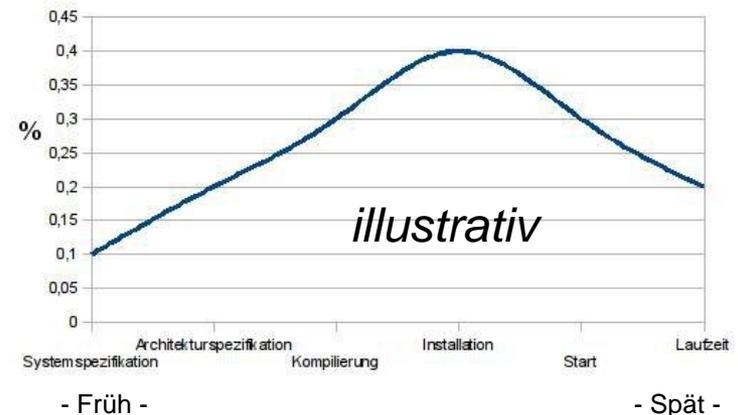
2. Software Produktlinie - Bindung -

- Bindung
 - Variabilitätspunkt wird mit Variante besetzt
- Bindungspunkte
 - Entscheidung „**wie**“ im Code/Deployment eine Variante gewählt wird
 - Patterns: Dependencies, Properties, Runtime-Button
- Bindungszeitpunkte
 - Entscheidung „**wann**“ eine Variante gewählt werden kann
 - Vgl. fertiges Auto:
 - Ausstattungspakete wählen (beim Kauf)
 - Navi-Karten aktualisieren (vor Autostart)
 - Sport-Schaltung aktivieren (während Fahrt)
 - ESP deaktivieren (während Fahrt)

Übersicht Bindungszeitpunkte:

	flexibility	performance	code size	complexity
source time	-	+	+	-
compile time	+	+	+	-
link time	+	+	+	-
load time	++	+	+	+
run time	+++	-	-	+

Anteile genutzter Bindungszeitpunkte:



3. Entwicklungsumgebung

Worin ist die SPL realisiert?

Standardzeugs...

3. Entwicklungsumgebung - Low Impact -

- **OS:** Windows/Linux
- **Versionsverwaltung:** SVN
- **Sprache:** Java 6 (Java 7 kompatibel)
- **IDE:** Eclipse + Plugins
- **Build & Dependency Mgmt:** Ant + Ivy (+ Groovy)

Wieso kein Maven?

→ SPL erfordert abweichende Konventionen

▪ Dokumentiert:



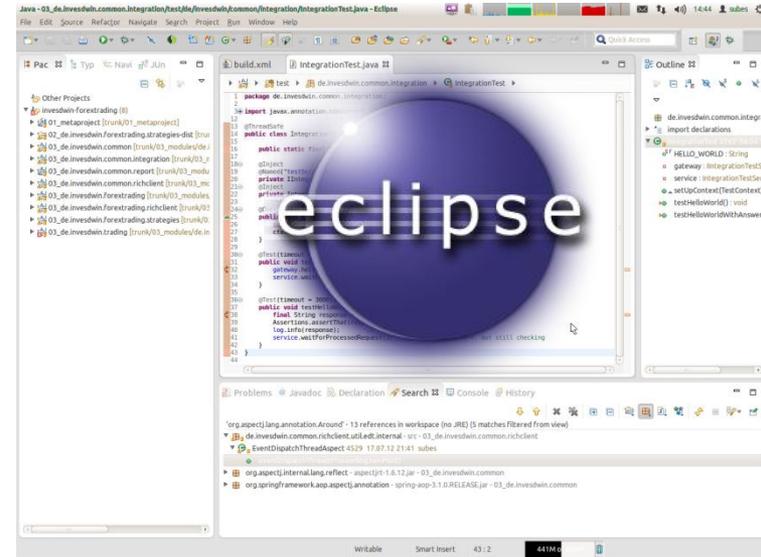
Einrichtung der Entwicklungsumgebung

Wir nutzen immer sofern möglich die aktuellste Version der jeweils eingesetzten Software. Die wichtigsten Grundkomponenten sind Java und Eclipse. Wie ein System auf dieser Basis zu konfigurieren ist, wird in diesem Artikel beschrieben. Die Software ist in der Regel so ausgewählt, dass sie sowohl unter Windows als auch Linux als Umgebung zur Verfügung steht. Empfohlen wird jedoch Linux, daher beschreibt dieser Artikel wie man sich Ubuntu als empfohlene Umgebung einrichtet.

Aus folgenden Bestandteilen besteht die Entwicklungsumgebung:

- **Umgebung:**
 - Ubuntu
 - alternativ kann man auch MS Windows verwenden (mit eigener Lizenz)
 - VirtualBox (optional)
 - alternativ könnte man auch den kostenfreien VMWare-Server nutzen
- **Office:**
 - LibreOffice
 - man kann für extreme Fälle auch MS Office in einer VM greifbar halten (mit eigener Lizenz)
 - Thunderbird (optional)
 - in Kombination mit einem GMail Account. Es gibt hierzu ein paar nützliche Plugins die man installieren sollte
 - OpenProj (optional)
 - die äquivalente gute kostenfreie Alternative zu MS Project
 - Mind (optional)
 - damit kann man schöne Mind Maps erstellen
- **Entwicklung:**
 - RabbieVCS
 - alternativ nutzt man unter Windows TortoiseSVN
 - MySQL
 - zurzeit keine Alternative möglich
 - OpenJDK
 - alternativ kann man auch das offizielle Oracle/Sun JDK verwenden
 - Eclipse
 - keine Alternative möglich
 - RabbitMQ (optional)
 - als präferierte Messaging Lösung
 - TextLive (optional)
 - mit Text Live unter Linux oder MikTex unter Windows als Backend
 - StatET (optional)
 - mit OR als Backend um statistische Auswertungen zu machen; alternativ auch Octave als vollwertiger Ersatz für Matlab
 - Maven (optional)
 - wir nutzen zwar Ivy in der Produktlinie, jedoch benötigt man es für öffentliche Libraries an denen wir auch ab und zu etwas machen müssen

WARNUNG: Bei dieser Softwareauswahl handelt es sich um Lizenztechnisch unbedenkliche Lösungen. Falls ein Entwickler etwas anderes unter eigener Lizenz verwenden möchte, tut er dies auf eigene Verantwortung und Gefahr.



Plugins

Eclipse benötigt einige Plugins, um damit arbeiten zu können. Installieren tut man diese entweder über den Eclipse Marketplace, über Update-Sites oder als Dropin. Empfohlen werden die Plugins auch in genau dieser Reihenfolge. Je nachdem, welche der Methoden zur Verfügung stehen, wird diese beim jeweiligen Plugin genannt. Falls sich eine Update-Site mittlerweile als unpassend erweist, da sie nicht mehr zur aktuellen Eclipse-Version passt. Kann man sich den aktuellen Link von der Homepage herausuchen und hier im Artikel updaten. Falls mittlerweile eine der bevorzugteren Installationsmethoden zur Verfügung steht, sollte dies am besten auch aktualisiert werden. Gelegentlich kann es auch vorkommen, dass die Installation über den Eclipse Marketplace nicht funktioniert, in solchen Fällen muss sich auch über alternative Installationsmethoden auf der Webseite des Plugins informieren.

Subversion

Mit diesem Plugin erhält man Zugriff auf SVN-Operationen von Eclipse heraus. Es sorgt auch dafür, dass das lokale Repo synchron zu den in Eclipse hinzugefügten und gelöschten Dateien bleibt.



Modul: Subversive - SVN Team Provider

Konfiguration

- Folgende Einstellungen sollte man am Plugin vornehmen:
1. Nach der Installation wird man gefragt, welche Konnektoren man installieren möchte. **SVN Kit** ist mittlerweile recht ausgereift, daher kann man sich die aktuellste Version von diesem Konnektor installieren.
 2. Unter **Window** -> **Preferences** -> **Team** -> **SVN** -> **Performance** wählt man **Calculate precise Team Menu enablements** an
 3. Unter **Window** -> **Preferences** -> **Team** -> **SVN** -> **Label Decorations** -> **Icon Decorations** wählt man alles an
 4. Unter **Window** -> **Preferences** -> **Team** -> **SVN** -> **Label Decorations** -> **Text Decorations** fügt man **{author}** am Ende zu **File** und **Folder** hinzu und leert das Feld für **Outgoing flag** und **Added flag**.
 5. Falls ein Projekt nicht automatisch als SVN-Projekt erkannt wird, kann man es mittels **Recht Maustaste auf das Projekt** -> **Team** -> **Share Projekt** -> **SVN** -> ... richtig konfigurieren. Alternativ könnte es auch klappen, das Projekt aus dem Workspace zu entfernen und erneut zu...

st meist die Integration von Subversion zum Gnome-Keyring verantwortlich und muss 'subversion/config' folgende Zeile:

ausprobieren:

3. Entwicklungsumgebung - Voraussetzungen -

- **Rechner:**

- **CPU:** Dual Core
- **RAM:** 4 GB
- **Festplatte:** nicht zu langsam
- **Speicherplatz:** ~3GB frei
- **Build-Zeiten:** Anhand Plattformprojekt
 - ~ 8 Minuten für 18 Module
 - => ~ 30 Sekunden pro Modul
 - (Halbierung durch bessere Hardware möglich)*



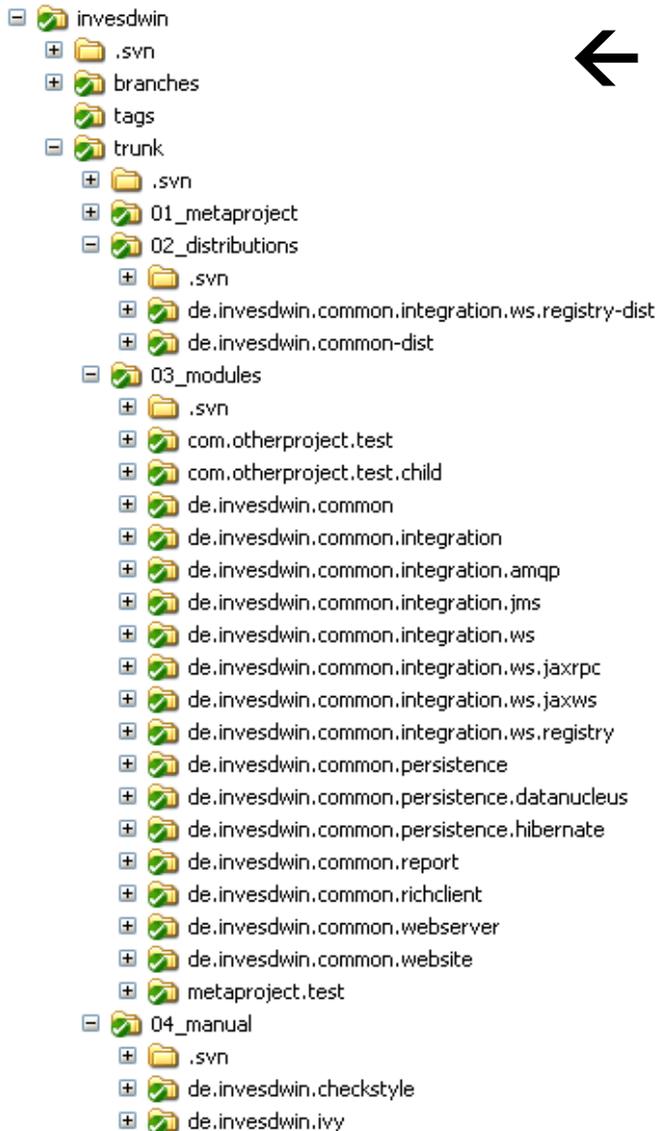
- **Know-How:**

- JEE-Erfahrung erforderlich (z.B. bisherige Plattform)
- Grundlegende Kenntnisse zu Spring wünschenswert
- Nach 1-2 Wochen Entwicklung damit sollte jemand in der Plattform gut klarkommen

4. Konfigurationsmanagement

Ein wissenschaftlich neuer Ansatz

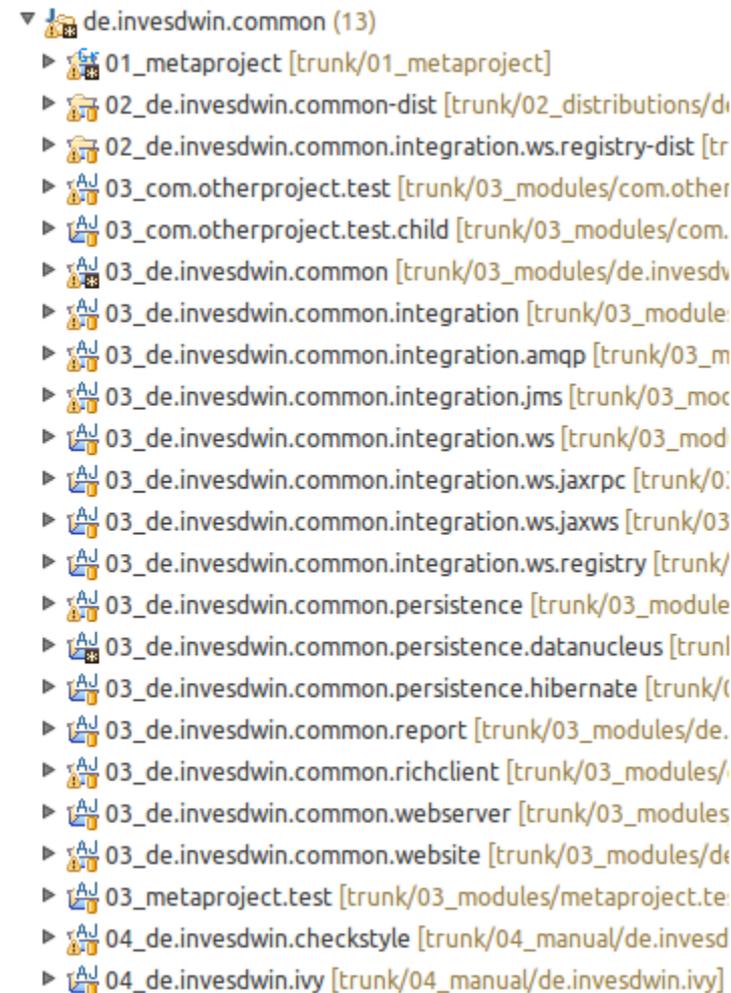
4. Konfigurationsmanagement - Übersicht -



← Explorer
Eclipse →

Produkt:

- 01 Metaprojekt
- 02 Distribution
- 03 Modul
- (04 manuelles Modul)



4. Konfigurationsmanagement

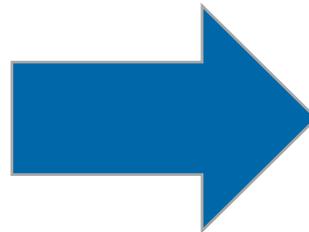
- Modul I -

- **Wiederverwendbarer Baustein für Produkte**
 - **Technisch:** Best Practices, Patterns, Frameworks, Utils, Tools,
 - **Fachlich:** Services, Entitäten, Logisches, Algorithmen
- Jeweils ein eigenes Eclipse-Projekt, um Modularität zu gewährleisten

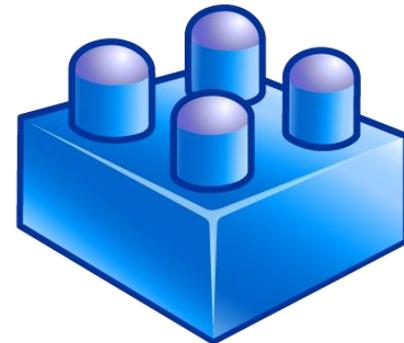


Komponenten

Anbieter, Vertrag, Kunde, ...



Zusammenfassen &
Wiederverwendbar machen



Modul

Vertragsverwaltung

4. Konfigurationsmanagement

- Modul II -

- Vergleich zur bisherigen Plattform:
 - **Modul > Container**
 - **Upgrade-Pfade** statt Copy-Paste
 - Lieber Variabilität zu Modul hinzufügen, statt neues Modul anlegen
 - Ziel: Effektiv weniger Wartungsaufwand

mehrmals unterschiedlich Fixen

VS

1 mal Fixen + mehrmals Version aktualisieren

- Lessons Learned aus alten Business Components anwenden:
 - **YAGNI & KISS**
 - Nicht auf der **grünen Wiese** entwickeln und **Over-Engineering** vermeiden
 - Variabilität erst hinzufügen, wenn sie benötigt wird
 - **Lose Kopplung** forcieren
 - Submodule nutzen, um Funktionalität/Alternativen zu kapseln

4. Konfigurationsmanagement

- Modul III -

▪ Wieso kein OSGI?

- Dynamisches Load/Unload unnötig
- OSGI-Deskriptoren schlecht gepflegt in Open Source Projekten
- Classpath-Probleme vermeiden (ein gemeinsamer Classpath ist einfacher)
- Jar-Hell und Versionskonflikte bereits durch Dependency Management gelöst
- Lose Kopplung durch „internal“ Packages mit Checkstyle-Prüfung gewährleistet
 - begründete Abweichung der „internal“ Package Regel möglich
 - ohne technische Herausforderungen



nur
OSGI
Kompatibel

4. Konfigurationsmanagement - Distribution -

- Ein konfiguriertes, deploybares Produkt

- Zusammenfassung mehrerer Module
- **Konfiguration der Variabilität** über
 - Dependencies
 - Properties
 - Zusätzliche Ressourcen
- Zielkundenspezifisch
- Zielumgebungsspezifisch

- Entscheidung über **Pakettyp** erst hier

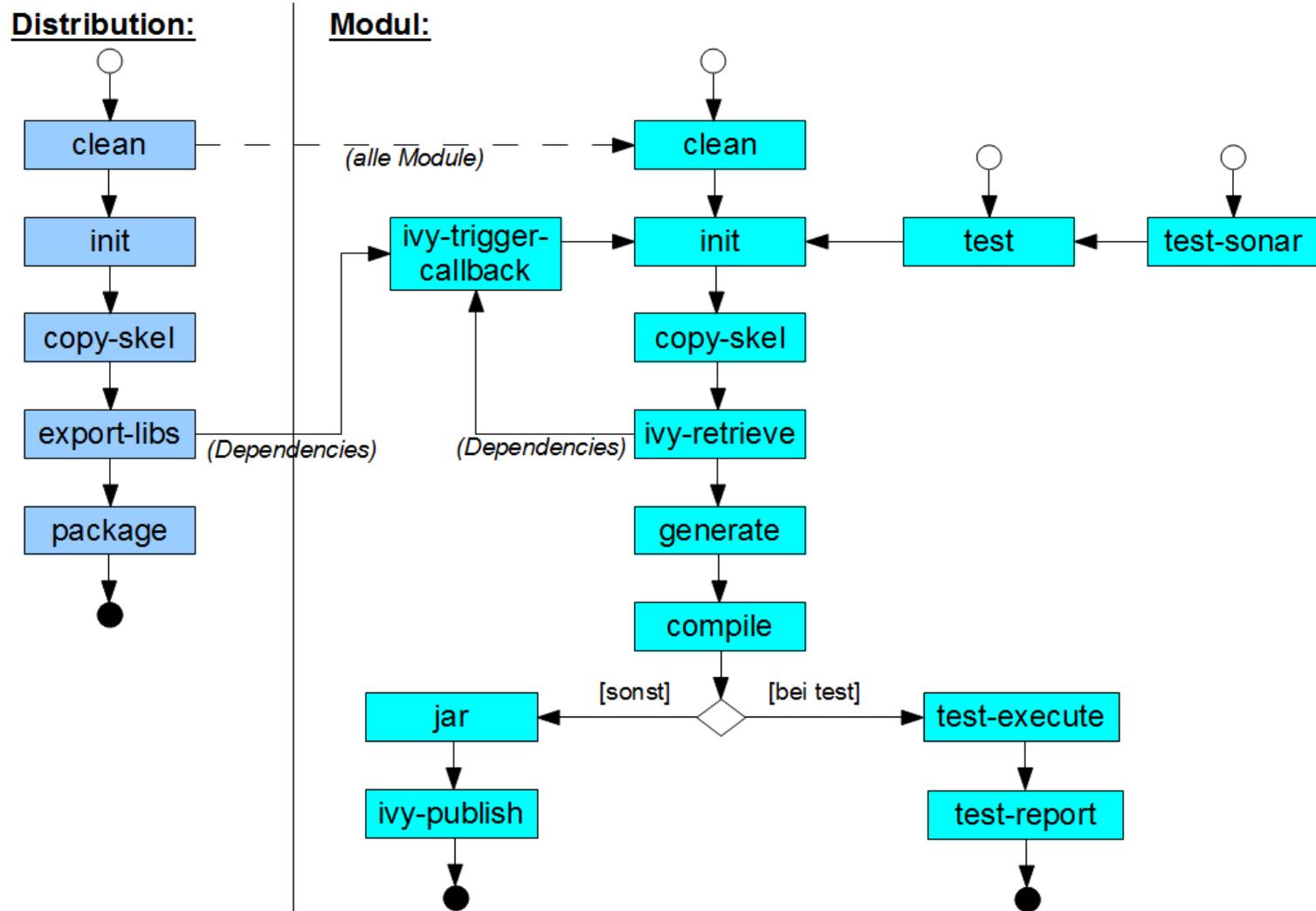
- Ausführbares **Fat-Jar**
- **Zip** mit Startskripten
- **War** als Container

→ Beispiel Webanwendung: Fat-Jar mit embedded Jetty **oder** War für Tomcat?



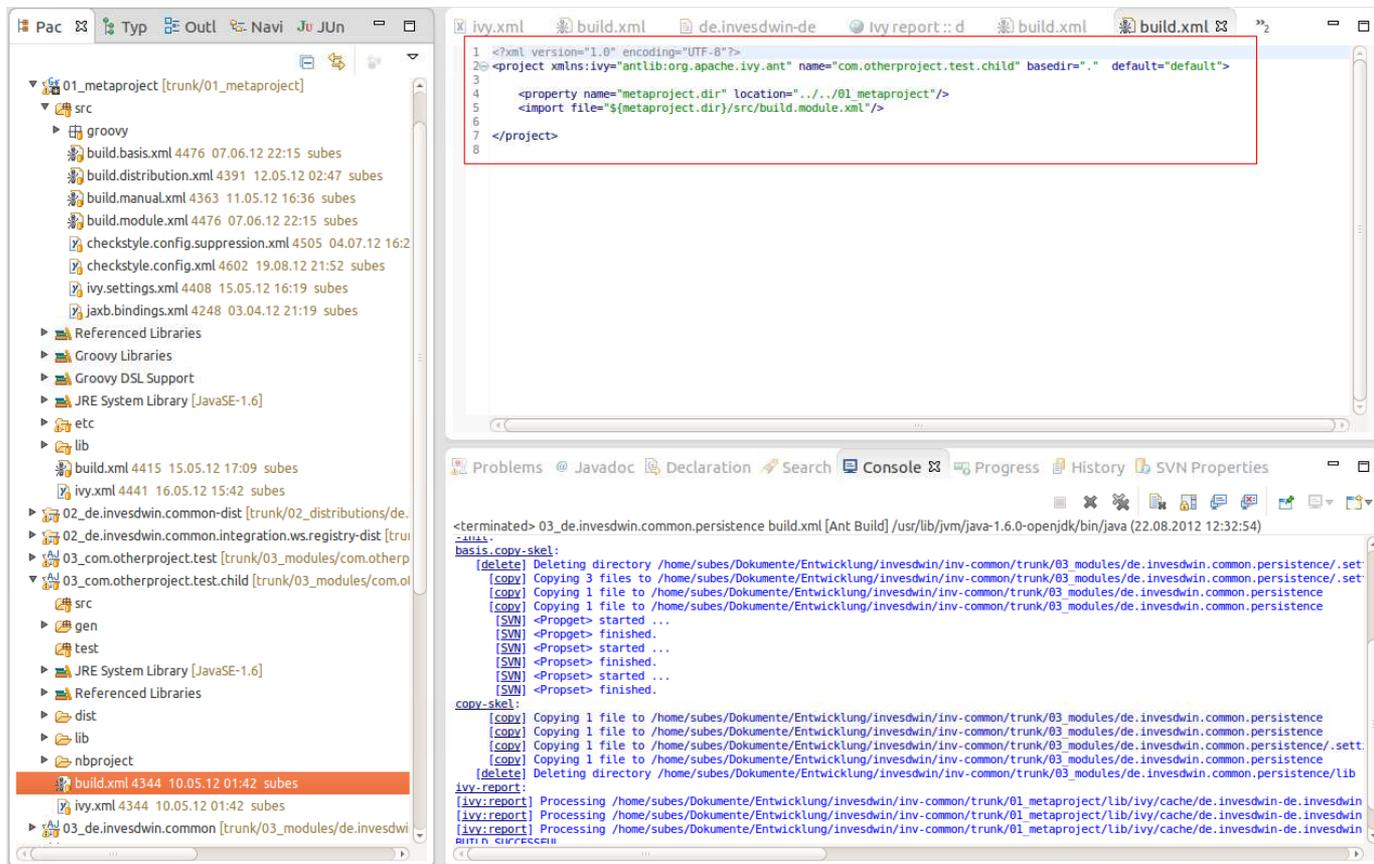
4. Konfigurationsmanagement - Metaprojekt I -

- Definition des einheitlichen Build-Prozesses:



4. Konfigurationsmanagement - Metaprojekt II -

- Wiederverwendbare standardisierte Ant-Skripte → nur an einer Stelle pflegen
- Projektvorlagen für Module und Distributionen → à la Maven Archetype
- Andere Assets erben Eclipse-Projekt-Konfiguration → nur einmal Konfigurieren



4. Konfigurationsmanagement - Metaprojekt III -

■ Automatisch generierter Eclipse-Classpath für Dependencies:

- Projekt-Referenzen für modulübergreifende Refactorings
- Einheitliche Benennung der Dependencies (<Modul>-<Version>.jar)
- Sources zu allen Jars automatisch angehängt
- AspectJ und weitere Eclipse-Plugins automatisch konfiguriert

→ Reduktion des Pflegeaufwandes für Module, lediglich Konfiguration in **ivy.xml** notwendig!

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <ivy-module version="2.0" xmlns: xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation="http://ant.apache.org/ivy/schemas/ivy.xsd"
4   xmlns="http://ant.apache.org/ivy/maven">
5   <info
6     organisation="de.invesdwin"
7     module="${ant.project.name}"
8     status="release"
9     revision="${ivy.version.invesdwin}"
10  </info>
11  <<configurations>
12    <conf name="default" />
13    <conf name="tests" />
14    <conf name="sources" />
15  </configurations>
16  <<publications defaultconf="default">
17    <artifact conf="default" type="jar" ext="jar" />
18    <artifact conf="tests" type="jar" ext="jar" m:classifier="tests" />
19    <artifact conf="sources" type="source" ext="jar" m:classifier="sources" />
20  </publications>
21  <<dependencies>
22    <!-- spring orm -->
23    <dependency org="org.springframework" name="spring-orm" rev="${ivy.version.spring}" conf="default->master;sources" />
24    <dependency org="org.springframework" name="spring-jdbc" rev="${ivy.version.spring}" conf="default->master;sources" />
25    <dependency org="com.mysma.querydsl" name="querydsl-jpa" rev="${ivy.version.querydsl}" conf="default->master;sources" />
26
27    <!-- connection pooling -->
28    <dependency org="com.jolbox" name="bonecp" rev="${ivy.version.bonecp}" conf="default->master;sources" />
29
30    <!-- jpa metamodel generator -->
31    <dependency org="org.hibernate" name="hibernate-jpamodelgen" rev="${ivy.version.hibernate.jpamodelgen}" conf="default->master;sources" />
32
33
34    <!-- Test Konnektoren für Datenbanken -->
35    <dependency org="com.h2database" name="h2" rev="${ivy.version.h2}" conf="tests->master;sources" />
36    <dependency org="mysql" name="mysql-connector-java" rev="${ivy.version.mysql-connector}" conf="tests->master;sources" />
37
38    <dependency org="de.invesdwin" name="de.invesdwin.common" rev="${ivy.version.invesdwin}" conf="default;tests;sources" />
39  </dependencies>
40 </ivy-module>
```

de.invesdwin.common.persistence 0.2.0 by de.invesdwin
resolved on 2012-08-22 12:32:58

default tests sources

Dependencies Stats

#modules	59
revisions	59 (0 searched, 0 downloaded, 0 evicted, 0 errors)
artifacts	59 (0 downloaded, 0 failed)
artifacts size	14252 kB (0 kB downloaded, 14252 kB in cache)

Dependencies Overview

Module	Revision	Status	Resolver	Default	Licenses	Size
de.invesdwin.common by de.invesdwin	0.2.0	release	temp	false		244 kB
hibernate-jpamodelgen by org.hibernate	1.1.1.Final	release	invesdwin	false	Apache License, Version 2.0	157 kB
bonecp by com.jolbox	0.7.2-SNAPSHOT	integration	repo	false		116 kB
querydsl-jpa by com.mysma.querydsl	2.3.0	release	invesdwin	false		94 kB
spring-orm by org.springframework	3.2.0.M1	release	invesdwin	false	The Apache Software License, Version 2.0	374 kB
spring-jdbc by org.springframework	3.2.0.M1	release	invesdwin	false	The Apache Software License, Version 2.0	394 kB

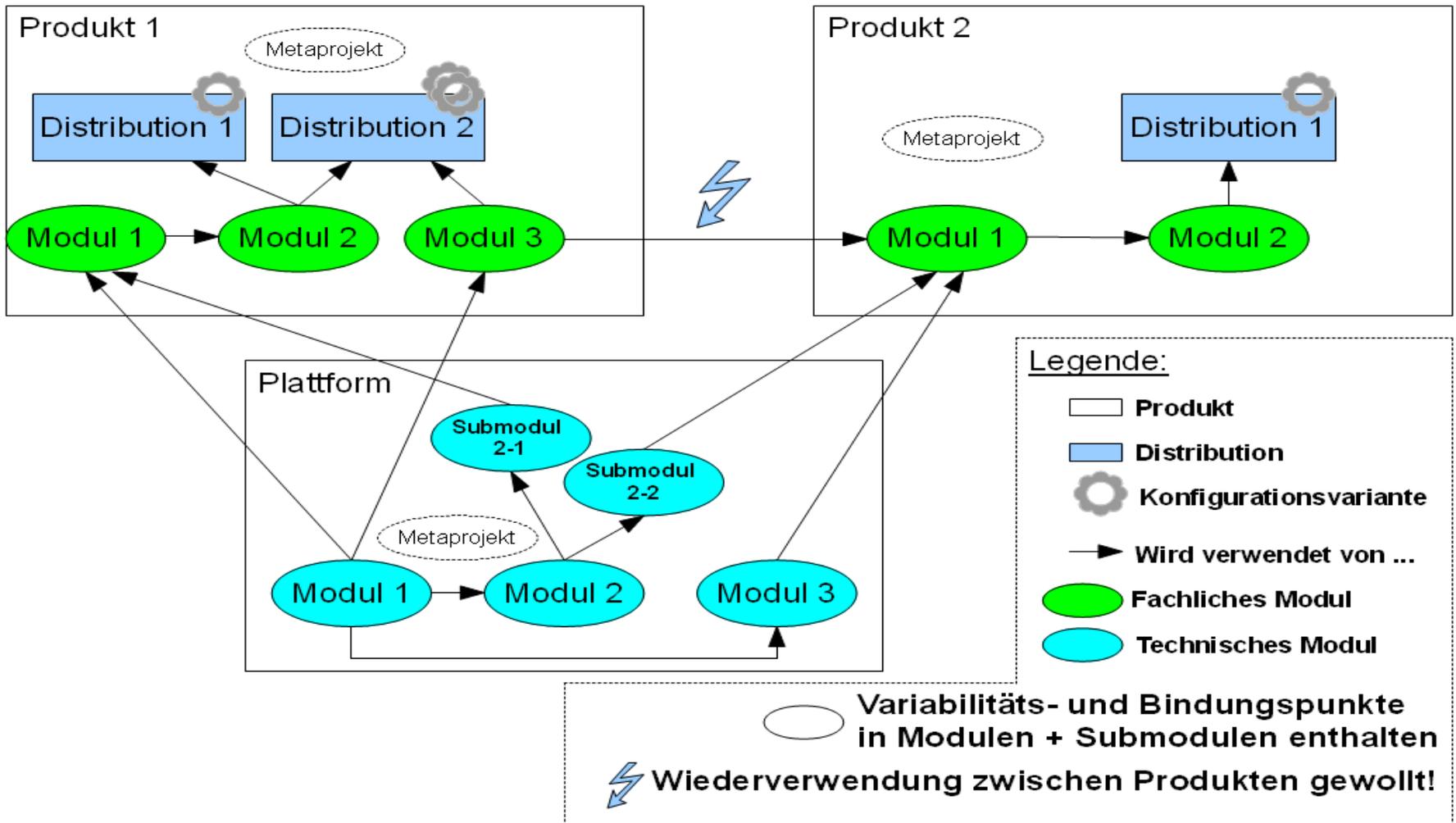
Details

de.invesdwin.common by de.invesdwin

Revision: 0.2.0

status	release
publication	20120809132657
resolver	temp
configurations	default

4. Konfigurationsmanagement - Multiproduktmanagement -



5. Assets

Was gibt es schon?

Plattform, Produkte, Bestandteile

5. Assets - Übersicht -



5. Assets

- Plattform I -

- metaproject



- investdwin-common



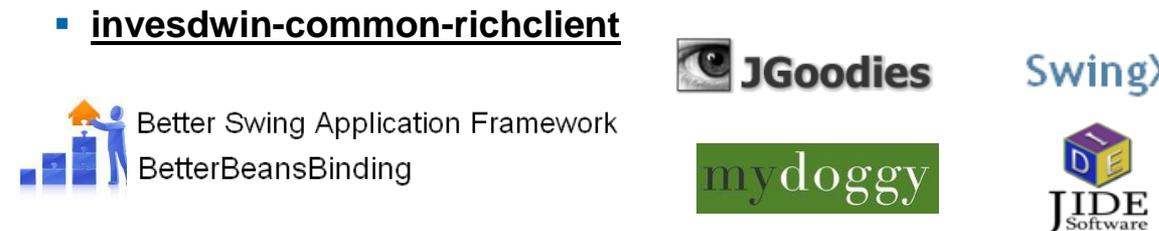
- investdwin-common-persistence



- investdwin-common-report



- investdwin-common-richclient



- Infrastruktur



- Dependency Repos



5. Assets

- Plattform II -

▪ investdwin-common-integration

- investdwin-common-integration-amqp



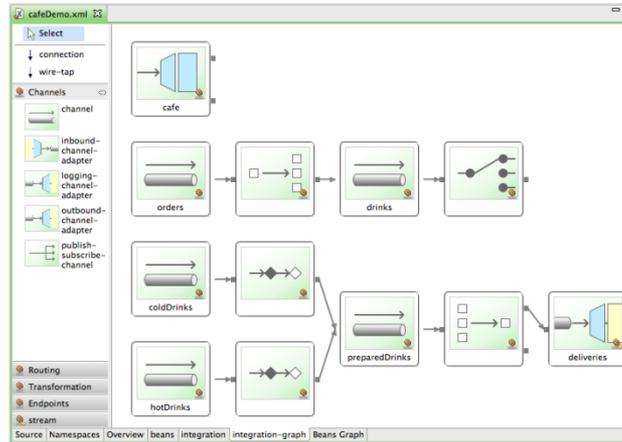
- investdwin-common-integration-jms



- investdwin-common-integration-ws



SPRING INTEGRATION



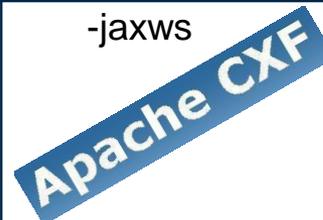
<?xml?>



-jaxrpc



-jaxws



-registry



▪ investdwin-common-website



▪ investdwin-common-webserver

Powered by



Deployment in Tomcat als WAR

konfigurierbar in Distribution



5. Assets

- Produkt webproxy -

- investdwin-webproxy(-...)

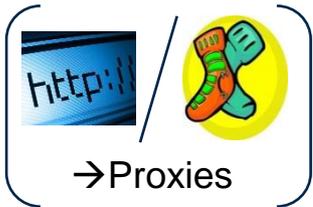


HtmlUnit

GUI-Less browser for Java programs



JAKARTA COMMONS
HTTPCLIENT



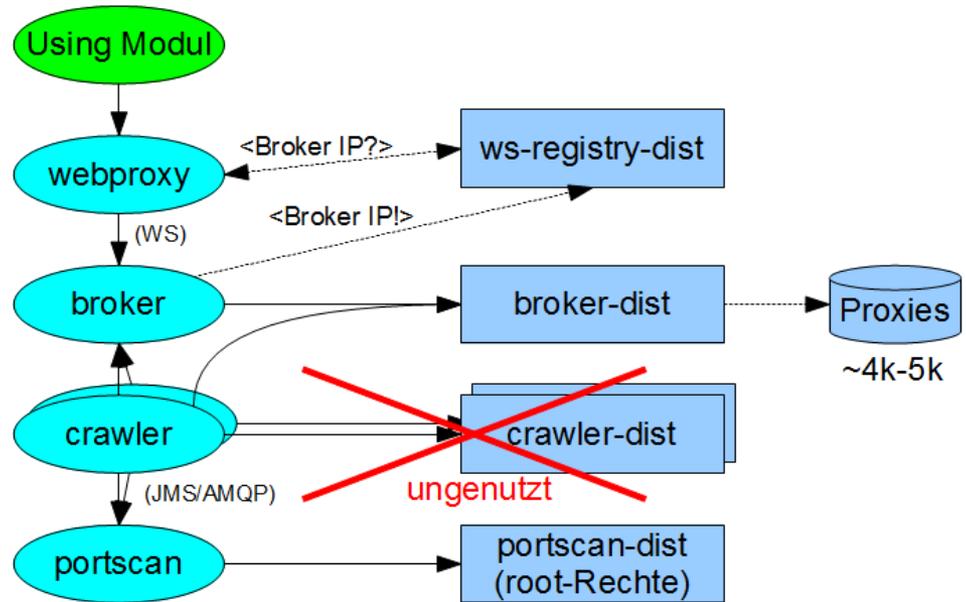
win/lib/j - **Peap**

→Proxies

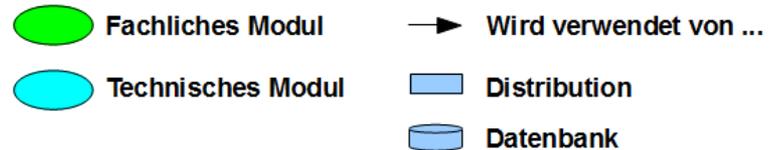


→Location

- Vereinfachtes Kontextdiagramm:



Legende:



6. Bindung der Variabilität

Bootstrap schafft Flexibilität

Pattern-Beispiele

6. Bindung der Variabilität - Application Bootstrap -

- Konfiguration verteilt über verschiedene Module
- Gemeinsamer Classpath (kein OSGI)
- Zwei Spring ApplicationContexte
 - Premerged → sammeln und konfigurieren
 - Merged → Anwendung laufen lassen
- Pro JUnit Testklasse ein neuer Bootstrap (schnell) für Konfigurationsänderungen:
 - Auswahl der Spring-XMLs mittels IContextLocation
 - Im Test mittels setUpContext()
 - Durch Mocks mittels ITestLifecycleHook

```
<terminated> 03_de.invesdwin.common [JUnit] /usr/lib/jvm/java-1.6.0-openjdk/bin/java (08.08.2012 18:55:17)
2012-08-08 18:55:21,369 [main] INFO d.i.common.beans.context.PreMergedContext.logLogbackBeingCon - Loading 2 logback configs from classpath [/META-INF/logback/common.logback.x
2012-08-08 18:55:21,740 [main] INFO de.invesdwin.common.CommonProperties.info - Loading 1 base package in classpath [de.invesdwin]
2012-08-08 18:55:22,479 [main] INFO d.i.common.beans.context.MergedContext.logContextsBeingLoade - Loading 1 spring context from classpath in TEST config [/META-INF/ctx.spring
2012-08-08 18:55:23,122 [main] INFO d.i.c.b.c.internal.EhCacheConfigurationMerger.logXmIsBeingLo - Loading 1 ehcache config from classpath [/META-INF/ehcache/default.ehcache.x
2012-08-08 18:55:23,634 [main] INFO d.i.common.beans.context.MergedContext.info - Bootstrap finished after: PT3.411.102.5275
2012-08-08 18:55:23,701 [main] INFO de.invesdwin.common.beans.context.TestTest.tearDown - 1. test finished after: PT0.023.404.9315
2012-08-08 18:55:23,730 [main] INFO de.invesdwin.common.beans.context.TestTest.tearDown - 2. test finished after: PT0.000.061.7405
2012-08-08 18:55:23,740 [main] INFO de.invesdwin.common.beans.context.TestTest.tearDown - 3. test finished after: PT0.000.041.9055
2012-08-08 18:55:23,900 [main] INFO d.i.common.beans.context.MergedContext.logContextsBeingLoade - Loading 1 spring context from classpath in TEST config [/META-INF/ctx.spring
2012-08-08 18:55:24,078 [main] INFO d.i.c.b.c.internal.EhCacheConfigurationMerger.logXmIsBeingLo - Loading 1 ehcache config from classpath [/META-INF/ehcache/default.ehcache.x
2012-08-08 18:55:24,119 [main] INFO de.invesdwin.common.lang.time.DurationTest.tearDown - 4. test finished after: PT0.000.099.7335
2012-08-08 18:55:24,134 [main] INFO de.invesdwin.common.lang.time.DurationTest.tearDown - 5. test finished after: PT0.010.332.3175
2012-08-08 18:55:24,139 [main] INFO de.invesdwin.common.lang.time.DurationTest.tearDown - 6. test finished after: PT0.000.065.9305
2012-08-08 18:55:24,172 [main] INFO de.invesdwin.common.lang.time.DurationTest.tearDown - 7. test finished after: PT0.027.914.4435
2012-08-08 18:55:24,269 [main] INFO d.i.common.beans.context.MergedContext.logContextsBeingLoade - Loading 1 spring context from classpath in TEST config [/META-INF/ctx.spring
2012-08-08 18:55:24,411 [main] INFO d.i.c.b.c.internal.EhCacheConfigurationMerger.logXmIsBeingLo - Loading 1 ehcache config from classpath [/META-INF/ehcache/default.ehcache.x
2012-08-08 18:55:24,601 [main] INFO de.invesdwin.common.beans.context.MockitoTest.tearDown - 8. test finished after: PT0.000.070.1215
2012-08-08 18:55:24,700 [main] INFO d.i.common.beans.context.MergedContext.logContextsBeingLoade - Loading 1 spring context from classpath in TEST config [/META-INF/ctx.spring
2012-08-08 18:55:24,854 [main] INFO d.i.c.b.c.internal.EhCacheConfigurationMerger.logXmIsBeingLo - Loading 1 ehcache config from classpath [/META-INF/ehcache/default.ehcache.x
2012-08-08 18:55:24,950 [main] INFO d.i.common.lang.math.scaled.PercentTest.tearDown - 9. test finished after: PT0.054.173.6455
```

6. Bindung der Variabilität - Properties -

- Alle Properties sind Systemproperties
- Damit verfügbar in:
 - anderen Properties Dateien à la Ant \${property}
 - XML (Spring, Frameworks mit Commons-Configuration)
 - Java (System.getProperty(„property“))
 - VisualVM (Monitoring, Vorsicht bzgl. Sicherheit!)

```
*PortscanProperties.java
1 package de.invesdwin.webproxy.portscan.internal;
2
3 import java.net.InetAddress;
4
5 @Immutable
6 public final class PortscanProperties {
7
8     public static final InetAddress CHECK_HOST;
9     public static final int CHECK_PORT = 80;
10    public static final int LOCAL_BIND_PORT;
11    public static final Duration ICMP_RESPONSE_TIMEOUT;
12
13    private static final SystemProperties SYSTEM_PROPERTIES = new SystemProperties(PortscanProperties.class);
14
15    static {
16        CHECK_HOST = NetworkUtil.toAddress(SYSTEM_PROPERTIES.getString("CHECK_HOST"));
17        LOCAL_BIND_PORT = readLocalBindPort();
18        ICMP_RESPONSE_TIMEOUT = SYSTEM_PROPERTIES.getDuration("ICMP_RESPONSE_TIMEOUT");
19    }
20
21    private PortscanProperties() {}
22
23    private static int readLocalBindPort() {
24        final String key = "LOCAL_BIND_PORT";
25        final Integer value = SYSTEM_PROPERTIES.getInt(key);
26        Assertions.assertThat(NetworkUtil.isPort(value))
27            .as(SYSTEM_PROPERTIES.getErrorMessage(key, value, null, "Value must be inclusively between "
28                + NetworkUtil.PORT_MIN + " and " + NetworkUtil.PORT_MAX + "."))
29            .isTrue();
30        return value;
31    }
32 }
```

```
de.invesdwin.webproxy.portscan.properties
1 #On the host port 80 must be open and a service has to be running on it. The host also has to answer pings so that the check
2 de.invesdwin.webproxy.portscan.internal.PortscanProperties.CHECK_HOST=google.de
3 de.invesdwin.webproxy.portscan.internal.PortscanProperties.LOCAL_BIND_PORT=44125
4 de.invesdwin.webproxy.portscan.internal.PortscanProperties.ICMP_RESPONSE_TIMEOUT=3 SECONDS
5 #For timings see: http://www.networkuptime.com/nmap/page09-09.shtml
6 de.invesdwin.webproxy.portscan.internal.PortscanProperties.UPLOAD_PAUSE_BETWEEN_PACKETS=0 MILLISECOND
7 de.invesdwin.webproxy.portscan.internal.PortscanProperties.UPLOAD_PAUSE_BETWEEN_PACKETS_PER_HOST=0 MILLISECOND
8 de.invesdwin.webproxy.portscan.internal.PortscanProperties.RESPONSE_TIMEOUT_BETWEEN_SYN_PACKETS_PER_HOST=500 MILLISECOND
9 de.invesdwin.webproxy.portscan.internal.PortscanProperties.MAX_OPEN_ICMP_REQUESTS=25
10 de.invesdwin.webproxy.portscan.internal.PortscanProperties.MAX_OPEN_SYN_REQUESTS=10
```

```
ctx.persistence.test.memory.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xmlns:context="http://www.springframework.org/schema/context"
4     xsi:schemaLocation="http://www.springframework.org/schema/beans
5         http://www.springframework.org/schema/beans/spring-beans.xsd">
6
7     <bean id="persistenceProperties" class="de.invesdwin.common.system.properties.SystemPropertiesDefinition">
8         <property name="systemProperties">
9             <map>
10                <entry key="javax.persistence.jdbc.driver" value="org.h2.Driver" />
11                <entry key="javax.persistence.jdbc.url" value="jdbc:h2:mem:invesdwin;DB_CLOSE_ON_EXIT=FALSE" />
12                <entry key="javax.persistence.jdbc.user" value="sa" />
13                <entry key="javax.persistence.jdbc.password" value="sa" />
14                <entry key="hibernate.dialect" value="org.hibernate.dialect.H2Dialect" />
15                <entry key="hibernate.hbm2ddl.auto" value="create" />
16            </map>
17        </property>
18    </bean>
19
20    <import resource="actx.persistence.hibernate.xml" />
21
22 </beans>
```

VisualVM Overview for de.invesdwin.webproxy.broker-prod-0.2.0.jar (pid 15284)

PID: 15284
Host: localhost
Main class: de.invesdwin.webproxy.broker-prod-0.2.0.jar
Arguments: <none>

JVM: OpenJDK 64-Bit Server VM (22.0-b10, mixed mode)
Java: version 1.7.0_03, vendor Oracle Corporation
Java Home: /usr/lib/jvm/java-7-openjdk-amd64/jre
JVM Flags: <none>

Heap dump on OOME: disabled

JVM arguments:

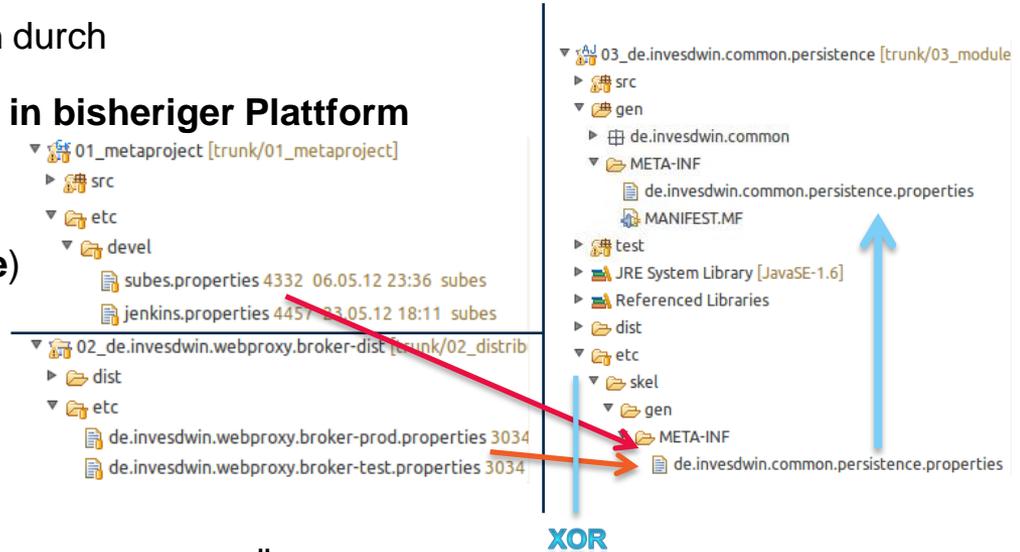
- de.invesdwin.webproxy.WebproxyProperties.PROXY_POOL_COOLDOWN_MIN_TIMEOUT=100 MILLISECOND
- de.invesdwin.webproxy.WebproxyProperties.PROXY_POOL_WARMUP_TIMEOUT=10 MINUTES
- de.invesdwin.webproxy.WebproxyProperties.PROXY_VERIFICATION_RETRY_SLEEP=15 SECONDS
- de.invesdwin.webproxy.WebproxyProperties.PROXY_VERIFICATION_RETRY_ON_ALL_EXCEPTIONS=false
- de.invesdwin.webproxy.broker.internal.BrokerProperties.ADDITIONAL_RANDOM_TO_BE_SCANNED_PORTS_PERCENT=25
- de.invesdwin.webproxy.broker.internal.BrokerProperties.MAX_SPECIFIC_TO_BE_SCANNED_PORTS=1000
- de.invesdwin.webproxy.broker.internal.BrokerProperties.PROXY_DOWNTIME_TOLERANCE=10 HOURS
- de.invesdwin.webproxy.crawler.internal.CrawlerProperties.RANDOM_SCAN_ALLOWED=false
- de.invesdwin.webproxy.crawler.internal.CrawlerProperties.WAIT_FOR_PORTSCAN_PROCESSING_END=true
- de.invesdwin.webproxy.geolocation.internal.GeolocationProperties.GEOIP_DATA_URL=http://geolite.maxmind.com/download/geoipt/
- de.invesdwin.webproxy.geolocation.internal.GeolocationProperties.GEONAMES_DATA_URL=http://download.geonames.org/export/
- ehcache.disk.store.dir=/tmp/15284@invesdwin.de/ehcache
- ehcache.disk.store.dir.persistent=/home/subs/invesdwin/cache/ehcache
- file.encoding=UTF-8
- file.encoding.pkg=sun.io
- file.separator=/
- hibernate.dialect=org.hibernate.dialect.MySQLInnoDBDialect
- hibernate.hbm2ddl.auto=update
- java.awt.graphicsenv=sun.awt.X11GraphicsEnvironment
- java.awt.printerjob=sun.print.PSPrinterJob

6. Bindung der Variabilität - Properties -

- **Default**-Werte in Modul-Properties und Ersetzen durch

Entwickler- und Distributions-Properties wie in bisheriger Plattform

- Build-Prozess **ersetzt** die Properties in referenzierten Modul-Projekten (**im gen-Source**) und geholten invesdwin-Libs (**im Jar**)



- **Variabler Bindungszeitpunkt** mit folgender Reihenfolge beim Überschreiben:

1. Source-Time: Default-Werte in Properties-Datei oder Spring-XML
2. Build-Time: Entwickler- oder Distributions-Properties
3. Load-Time: Spring-XML oder Java-Parameter via `-Dproperty=value`
4. Runtime: Java via `Properties.setProperty(„value“)`

	flexibility	performance	code size	complexity
source time	-	+	+	-
compile time	+	+	+	-
link time	+	+	+	-
load time	++	+	+	+
run time	+++	-	-	+

7. DDD und MDA

DDD!

Wo könnte man MDA ansetzen?

7. DDD und MDA

- Konzepte -

- **DDD – Domain Driven Design/Development**
 - Technische Module in Englisch
 - Fachliche Module in Domänensprache
 - Frontend internationalisiert → keine Modulkopie die übersetzt wird
- **MDA – Model Driven Architecture**
 - Bisher kein MDA:
 - Lieber Framework statt Generator → einfachere Wartung, simplerer Build-Prozess
 - Forschungsprojekt:
 - Kaum wiederkehrende Konzepte
 - Anfangsinvestition für Generator würde sich nicht auszahlen
 - MDA machbar:
 - Code-Generierung auf Modulebene, nicht übergreifend
 - Ein Modell pro Modul
 - Erweiterung des „generate“-Build-Schrittes
 - Viel SPL-Produktivitäts-Protenzial



8. Fazit & Diskussion

Vorteile

Nachteile

8. Fazit & Diskussion

- Eure Meinung ist sehr willkommen! -

Vorteile	Nachteile
+ SPL nicht mehr nur Theorie, sondern nutzbar	- Einarbeitungsaufwand
+ Hoher Entwicklungskomfort	- Neue Plattform
+ Hohe Flexibilität	- Neue Technologien
+ Günstigere Wartung bei komplexen Projekten und im Multiprojektumfeld	- Organisationsstrukturen ändern?
+ Crawler-Modul mit verbauter Tricktüte	- Mehr Diversität zu Managen
+ Deckt viele Features/Ansprüche von der bisherigen Plattform ab	
+ Standardisierte Eclipse-Projekte und Build-Skripte	
+ Offene, verbreitete Technologien	
+ Wiederverwendbarkeit als oberstes Ziel	



Vielen Dank für Ihre Aufmerksamkeit