

# CLEAN ARCHITECTURE

Systeme für die Ewigkeit

Lars Briem

(briem.lars@googlemail.com)

Duale Hochschule Baden Würtemberg - Standort Karlsruhe

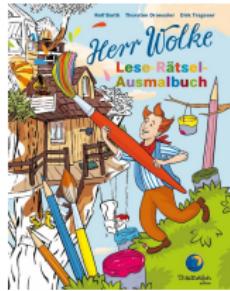
# Softwareentwicklung

- ▶ Beständiger Wandel alle fünf Jahre
  - ▶ Kein zentraler Takt
  - ▶ Unterschiedliche Zykluslänge von Produkten
- ▶ Keine monolithischen Systeme mehr
  - ▶ Zusammenstöpseln von Bausteinen (Building Blocks)
  - ▶ Fundament bildet häufig ein Framework



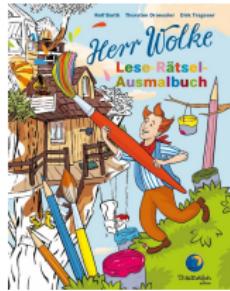
# Framework vs. Library

- ▶ Framework (Rahmenstruktur)
  - ▶ Semi-vollständige Anwendung
  - ▶ Kohärente Struktur
  - ▶ Entwickler vervollständigen *nur* die leeren Bereiche



# Framework vs. Library

- ▶ Framework (Rahmenstruktur)
  - ▶ Semi-vollständige Anwendung
  - ▶ Kohärente Struktur
  - ▶ Entwickler vervollständigen *nur* die leeren Bereiche



# Framework vs. Library

- ▶ Library (Programmbibliothek)
  - ▶ Sammlung von nützlichen Klassen und Methoden
  - ▶ Keine/kaum Anforderungen an Restprogramm
  - ▶ Keine Unterstützung für Strukturierung
  - ▶ Entwickler *kleben* Bibliotheken aneinander



# Framework vs. Library

- ▶ Frameworks binden die Anwender an sich
    - ▶ Starke Kopplung (Vendor lock-in)
    - ▶ Kopplung auch an den Lebenszyklus
  - ▶ Libraries lassen mehr Freiheiten
    - ▶ Starke Kopplung vermeidbar
- ⇒ Lieber Libraries als Frameworks verwenden
- ⇒ Frameworks nicht *wie gedacht* einsetzen

# Technologiewahl für Projekte

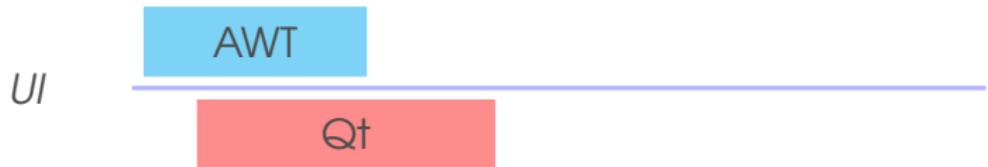
UI

---

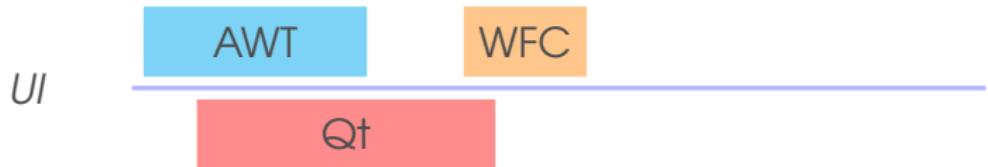
# Technologiewahl für Projekte



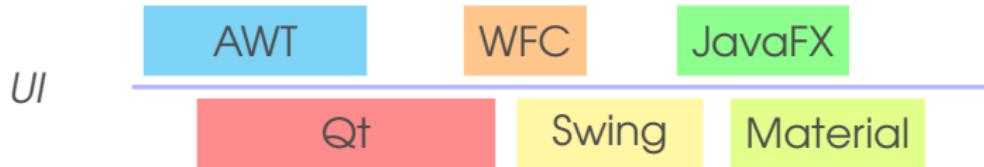
# Technologiewahl für Projekte



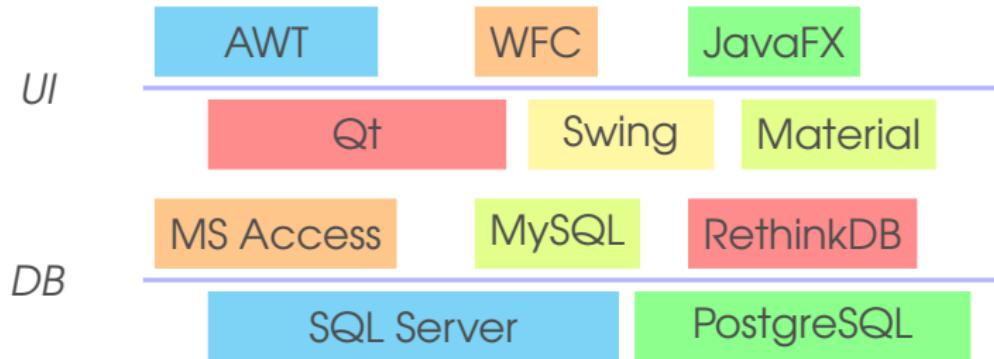
# Technologiewahl für Projekte



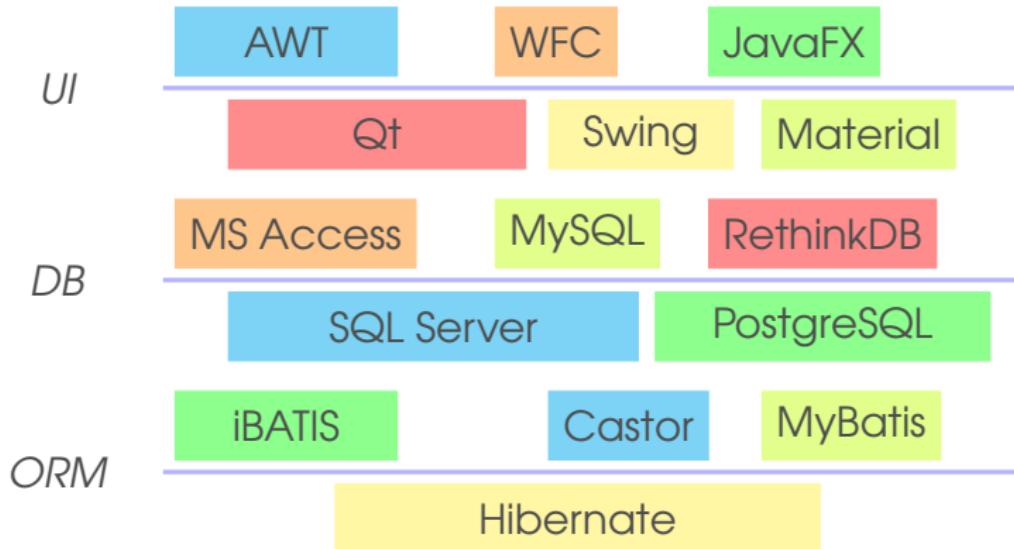
# Technologiewahl für Projekte



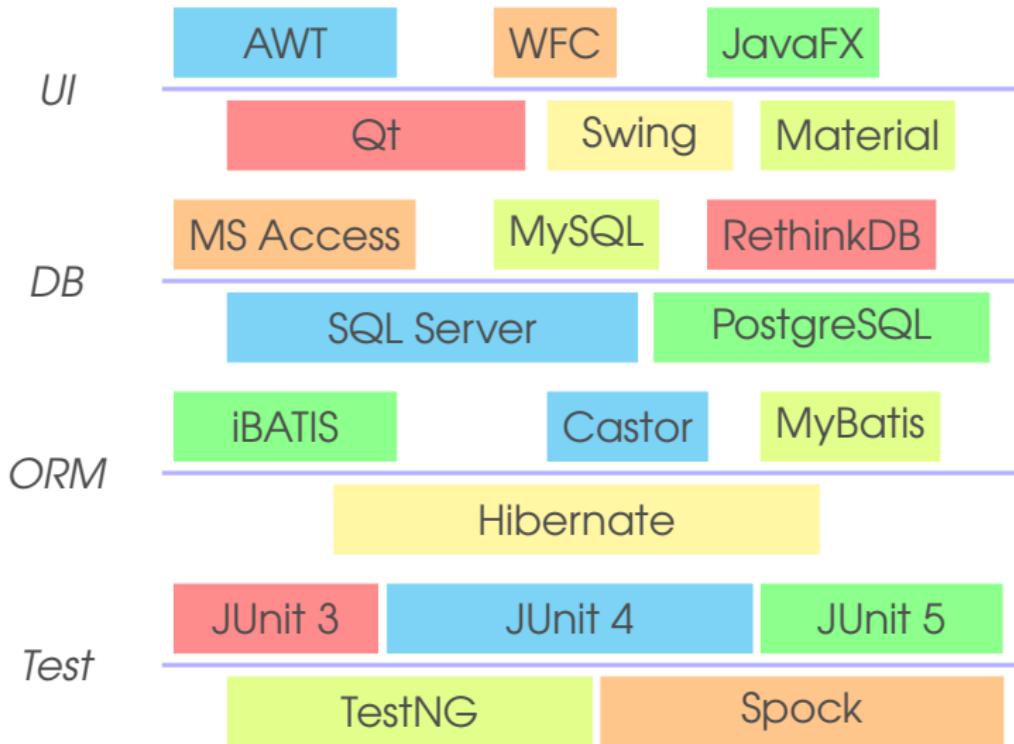
# Technologiewahl für Projekte



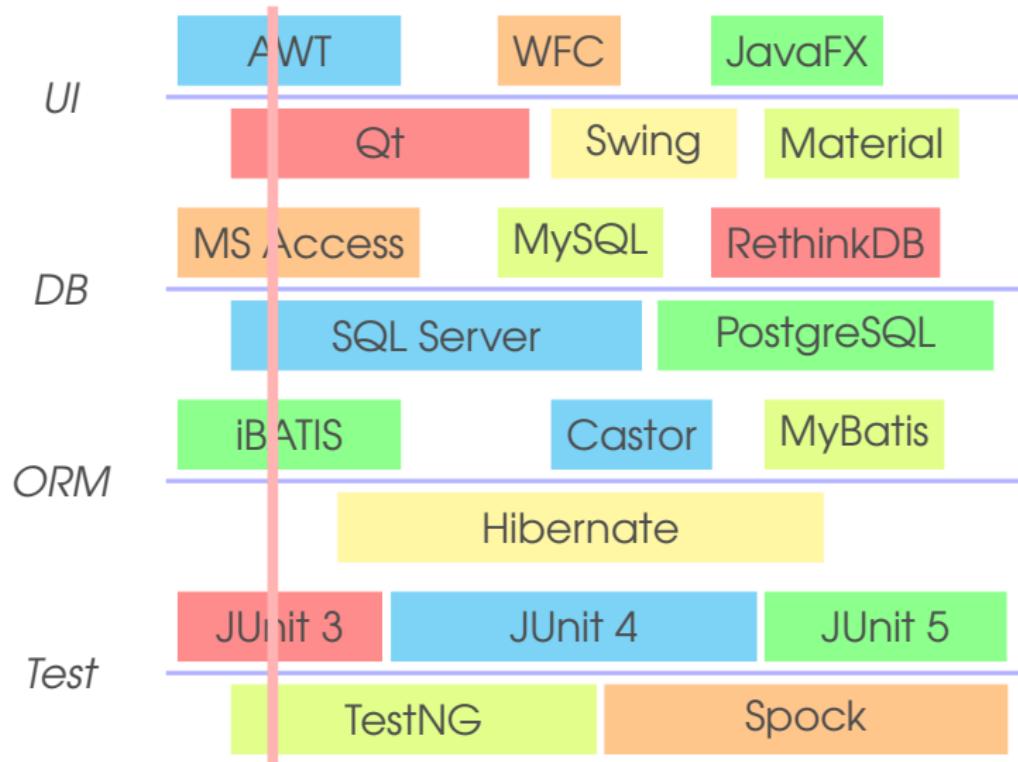
# Technologiewahl für Projekte



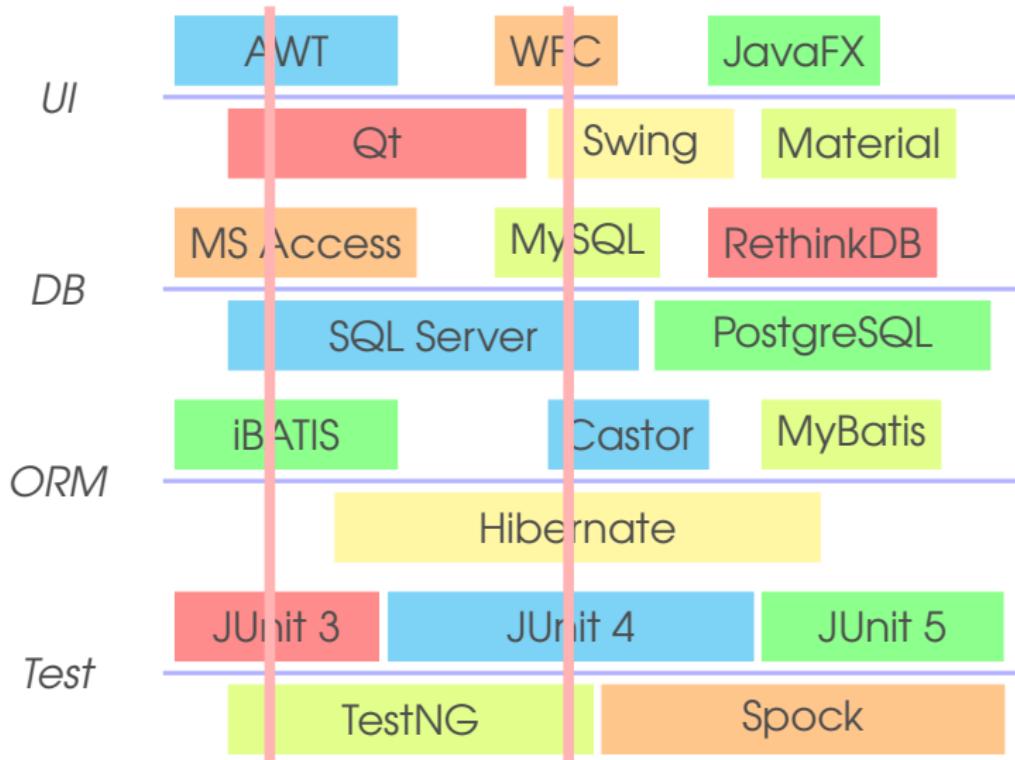
# Technologiewahl für Projekte



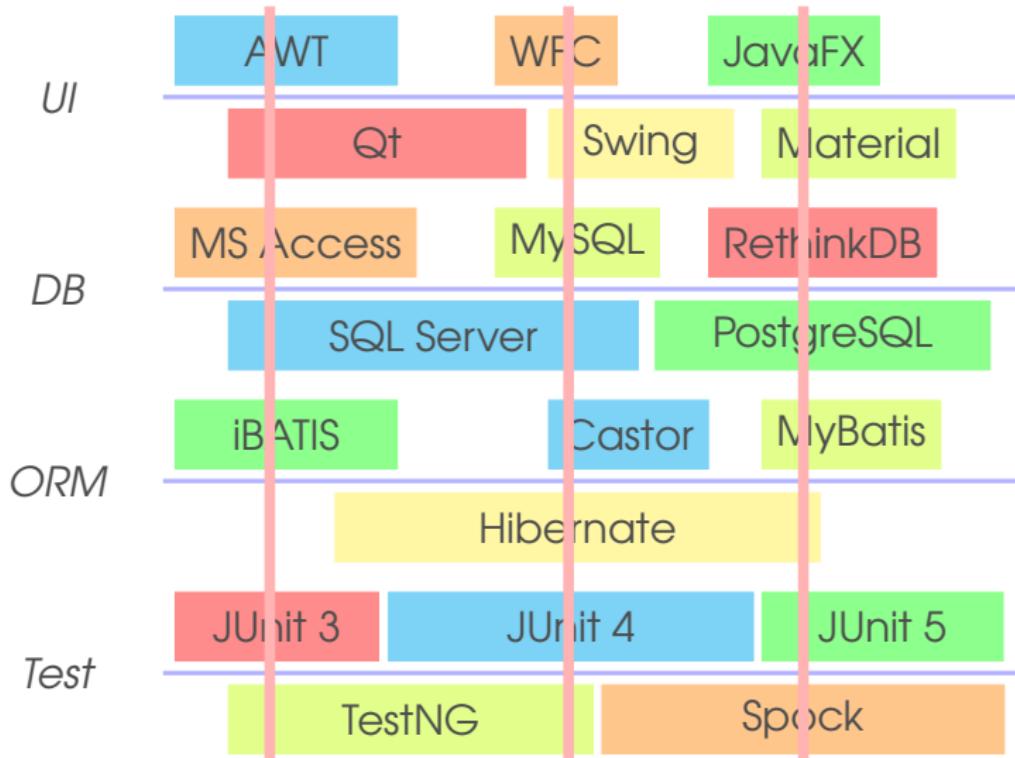
# Technologiewahl für Projekte



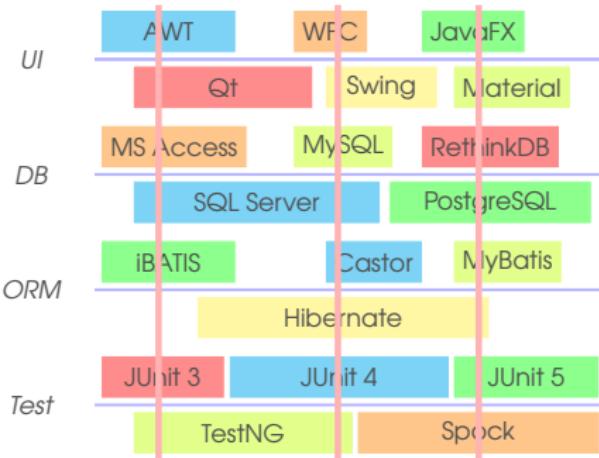
# Technologiewahl für Projekte



# Technologiewahl für Projekte



# Technologiewahl für Projekte



- ▶ Stark vom Zeitpunkt abhängig
  - ▶ Bei gleichen Anforderungen trotzdem unterschiedlich
  - ▶ Früh zu treffende Entscheidung
- ⇒ Immer ein Kompromiss

Tag 1	Tag 2	Tag 3
AWT	Swing	JavaFX
MS Access	MySQL	RethinkDB
iBATIS	Hibernate	MyBatis
JUnit	TestNG	Spock

# Nachhaltige Technologiewahl

- ▶ Gute Entscheidungen werden spät getroffen
- ▶ Strukturen (Architektur) so wählen, dass Entscheidungen verzögert werden können
  - ▶ Ohne negative Folgen
- ▶ Minimalziel: Entscheidungen revidieren können
  - ▶ Mit möglichst geringen negativen Folgen

# Kriterien einer nachhaltigen Architektur

- ▶ Eine langfristige Architektur ...
  - ▶ besitzt einen technologieunabhängigen Kern
    - ▶ Die eigentliche Anwendung
  - ▶ behandelt jede Abhängigkeit als temporäre Lösung
  - ▶ unterscheidet zwischen zentralem (langlebigem) und peripherem (kurzlebigerem) Sourcecode
- ▶ Metapher: Die Zwiebel
  - ▶ *Onion Architecture*



# Struktur der Clean Architecture



# Struktur der Clean Architecture



Bibliothek B  
(ORM)



Bibliothek A  
(GUI)

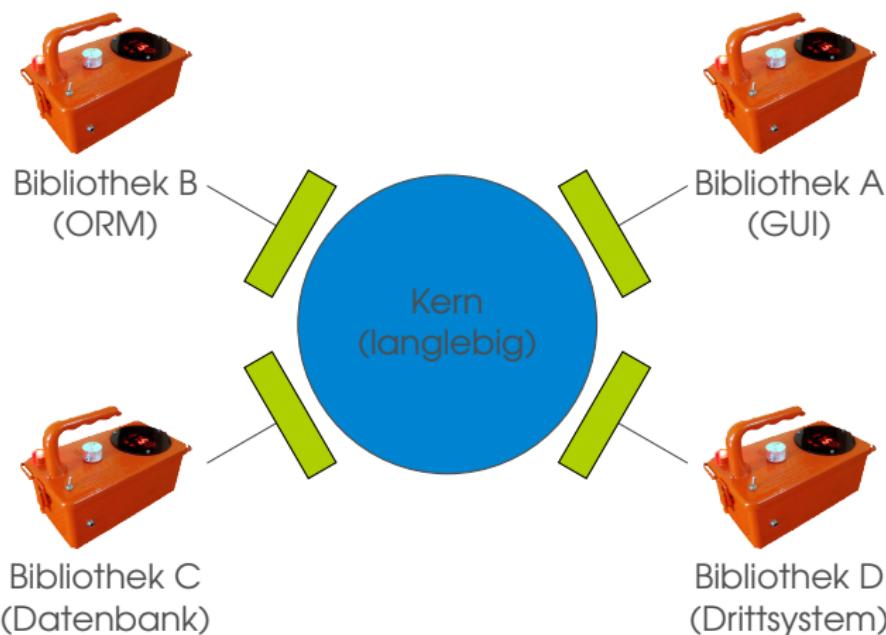


Bibliothek C  
(Datenbank)

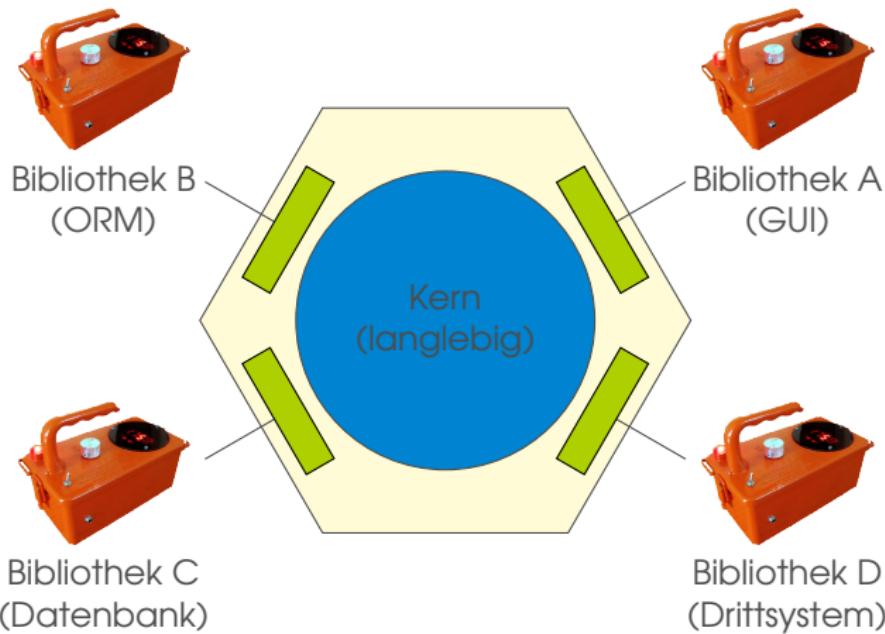


Bibliothek D  
(Drittsystem)

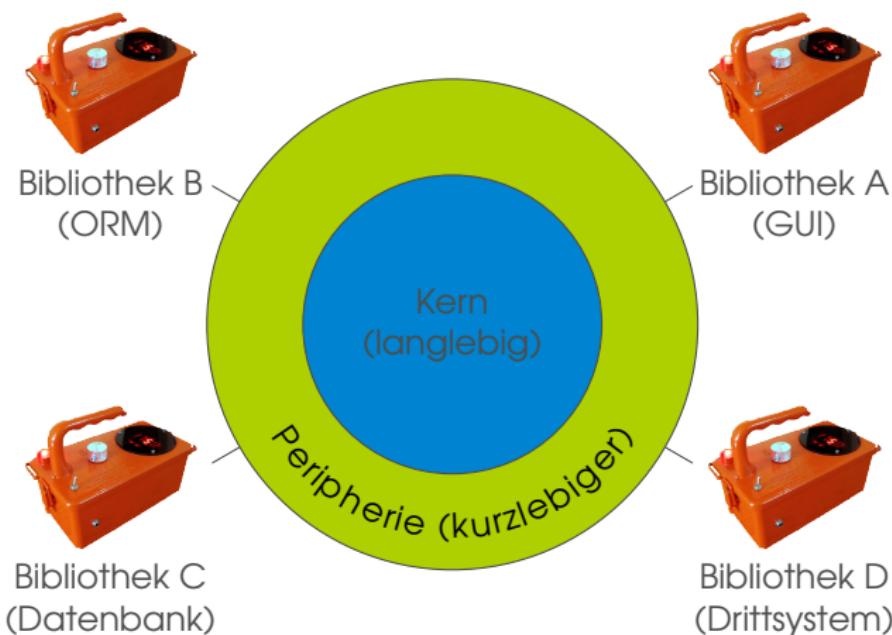
# Struktur der Clean Architecture



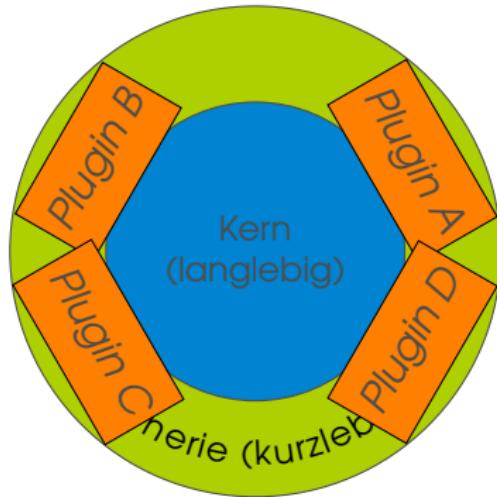
# Struktur der Clean Architecture



# Struktur der Clean Architecture



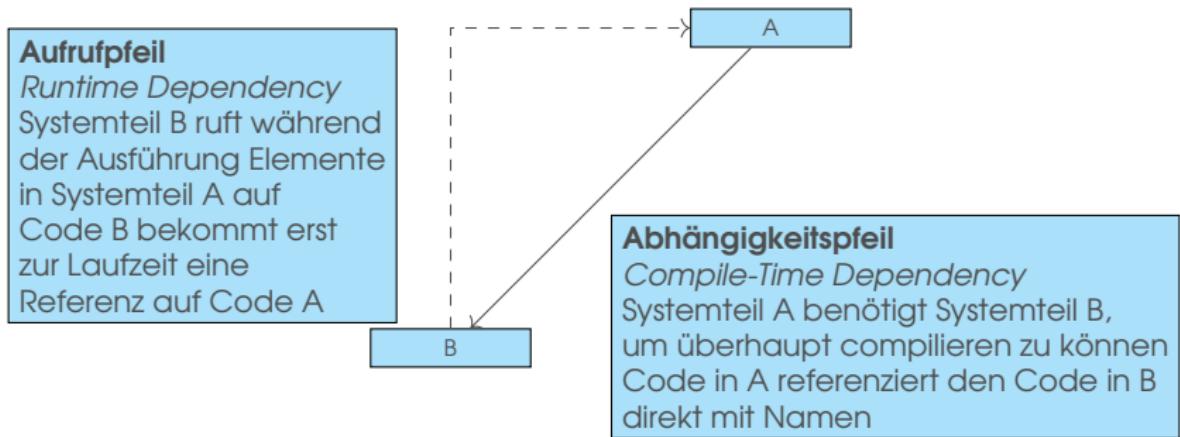
# Struktur der Clean Architecture



## Die Dependency Rule

- ▶ Zentrale Regel für Abhängigkeiten
  - ⇒ Abhängigkeiten immer von außen nach innen
- ▶ Erfordert für jede Klasse eine klare Positionierung
- ▶ Abhängigkeitspfeile gehen immer von außen nach innen
  - ▶ Aufrufpfeile können in beide Richtungen gehen

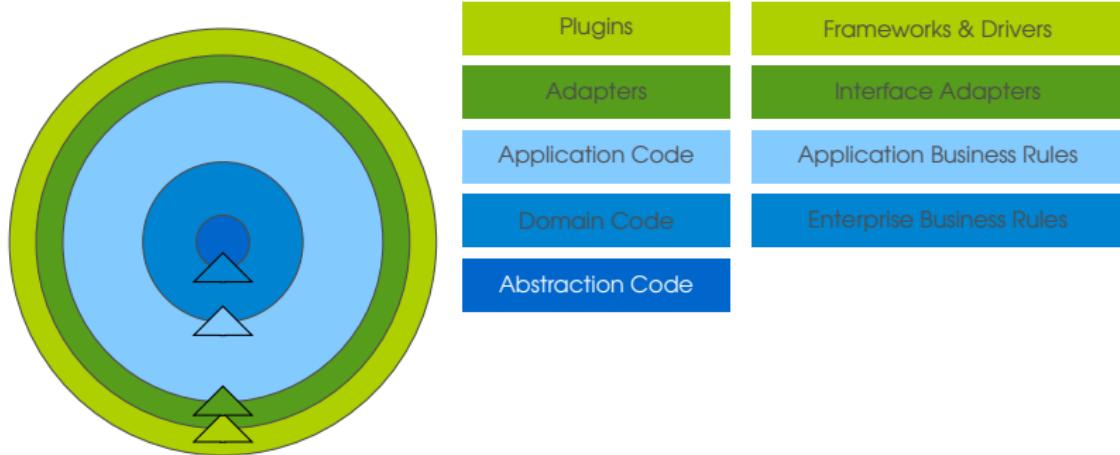
# Bedeutung der Pfeile



## Abhangigkeiten gestalten

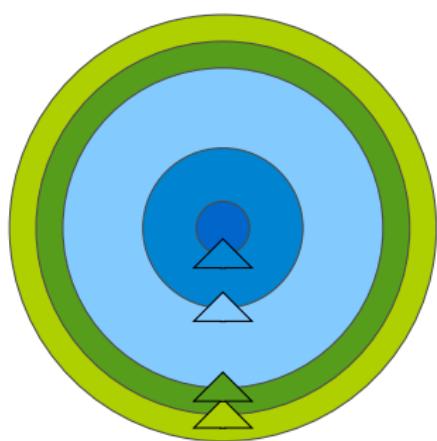
- ▶ Software-Architektur ist die Kunst, die Abhangigkeiten zwischen Systemteilen willentlich und zum Vorteil der Beteiligten zu gestalten
- ▶ Die Richtung und Art der Pfeile im Architekturdiagramm festzulegen, ist die Aufgabe des Software-Architekten
  - ⇒ Die Richtung kann beliebig gewahlt werden
  - ⇒ Wir konnen die Richtung jederzeit umdrehen!

# Struktur der Clean Architecture



- ▶ Innere Schichten wissen nichts von den Äußeren
  - ▶ Abhängigkeiten immer von außen nach innen
- ▶ Beliebig viele innere Schichten (oft drei)

# Position der Elemente



Plugins	Datenbank, GUI, Drittsysteme, Geräte
Adapters	Presenters, Controllers, Gateways
Application Code	Use Cases
Domain Code	Entities
Abstraction Code	Generic Entities (mathematische Konzepte)

- ▶ Die Position eines Elements wird durch seinen Einsatzzweck bestimmt
  - ▶ Nicht durch technische Bequemlichkeit
- ▶ Je konkreter (*low level*) der Code ist, desto weiter außen liegt er

# Grundregeln der Clean Architecture

- ▶ Der Anwendungs- und Domaincode ist frei von Abhängigkeiten
  - ▶ Sämtlicher Code kann eigenständig verändert werden
  - ▶ Sämtlicher Code kann unabhängig von Infrastruktur kompiliert und ausgeführt werden
- ▶ Innere Schichten definieren Interfaces, äußere Schichten implementieren diese
- ▶ Die äußeren Schichten koppeln sich an die inneren Schichten (Richtung Zentrum)

## Schicht 4: Abstraction Code

- ▶ Enthält domänenübergreifendes Wissen
  - ▶ Mathematische Konzepte (z.B. Matrizen)
  - ▶ Algorithmen und Datenstrukturen (z.B. Zelluläre Automaten)
  - ▶ Abstrahierte Muster (z.B. Quantitäten)
- ▶ Häufig nicht notwendig und/oder nicht vorhanden
- ▶ Wahrscheinlich bereits als Library verfügbar
- ▶ Kann nachträglich extrahiert werden
- ▶ Nicht aus Angeberei anlegen!

## Schicht 3: Domain Code

- ▶ Enthält v.a. Entities (Business Objects)
- ▶ Implementiert organisationsweit gültige Geschäftslogik (Enterprise Business Rules)
- ▶ Der innere Kern der Anwendung bzw. Domäne
- ▶ Sollte sich am seltensten ändern
  - ▶ Immun gegen Änderungen an Details wie Anzeige, Transport oder Speicherung
  - ▶ Unabhängig vom konkreten Betrieb der Anwendung
- ▶ Hohes emotionales Investment der Entwickler

### Enterprise Business Rules

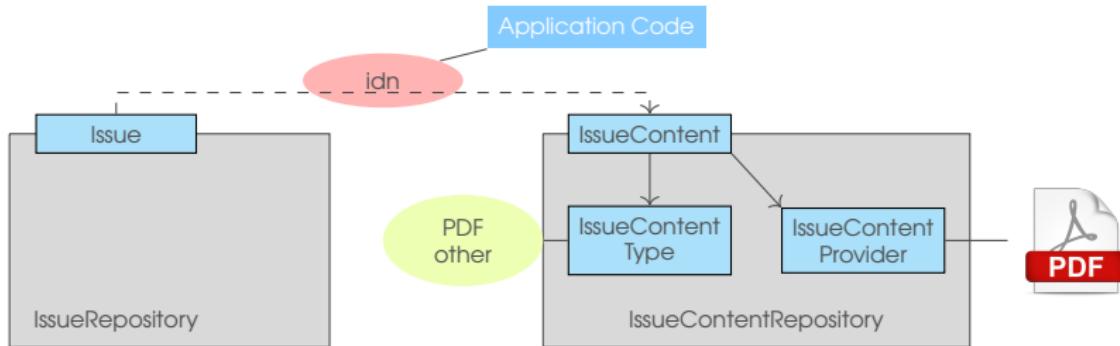
- ▶ Fachliche Regeln, die innerhalb einer Organisation immer gelten (vgl. Vorlesung DDD)
- ▶ Existieren immer, unabhängig davon, ob diese Regeln in einer Anwendung nachmodelliert wurden
- ▶ Beispiele
  - ▶ Ein Student kann genau eine Erstklausur, eine Nachklausur und eine mündliche Prüfung pro Modul ablegen
  - ▶ Ein Fahrzeug muss am Ende der Tour wieder nach Hause zurück gebracht werden

## Beispiel für Domain Code

- ▶ Domäne: Bankkonto-Verwaltungssoftware
- ▶ Ein zentraler Begriff ist das „Konto“
- ▶ Jedes Konto muss der zentralen Regel genügen:
  - ⇒ Die Summe der Zubuchungen, Abbuchungen und des inversen Kontostands ergibt immer 0
- ▶ Das Konto ist eine Klasse im Domain Code
- ▶ Die Regel ist eine Invariante in der Konto-Klasse
  - ▶ Jede Methode der Klasse Konto muss die Regel beachten

- ▶ Domain Driven Design (DDD) Aggregate sind typische Strukturen im Domain Code
  - ▶ Bestehen aus DDD Entities → Teil des Domain Code
  - ▶ Werden durch DDD Repositories verwaltet → Teil des Domain Code
- ▶ Weitere Bestandteile:
  - ▶ Verhalten der Entities
  - ▶ Verhalten im Aggregat
- ▶ Aggregat-übergreifendes Verhalten ist dann Bestandteil der Use Cases

# Beispiel für DDD und Domain Code



- ▶ Projekt zur Bereitstellung von E-Books
- ▶ Issue und IssueContent sind über die idn konzeptionell miteinander verbunden
- ▶ Diese Verbindung wird aber erst im Application Code umgesetzt

## Schicht 2: Application Code

- ▶ Enthält die Anwendungsfälle (Use Cases)
  - ▶ Resultiert direkt aus den Anforderungen
- ▶ Implementiert die anwendungsspezifische Geschäftslogik
  - ▶ Application-specific Business Rules
- ▶ Steuert den Fluss der Daten und Aktionen von und zu den Entities
  - ▶ Verwendet die Geschäftslogik, um den jeweiligen Anwendungsfall umzusetzen

## Schicht 2: Application Code

### Anwendungsspezifische Geschäftslogik

- ▶ Regeln, die nur für den Anwendungsfall gelten
  - ▶ Spezifisch für die Anwendung
  - ▶ Nicht organisationsweit gültig
  - ▶ z.B. Regeln für einen Workflow
- ▶ Beispiele
  - ▶ Ändern des Passworts erfordert Eingabe und Prüfung des alten Passworts
  - ▶ Reihenfolge von Wahlentscheidungen in der Verkehrsmodellierung

## Schicht 2: Application Code

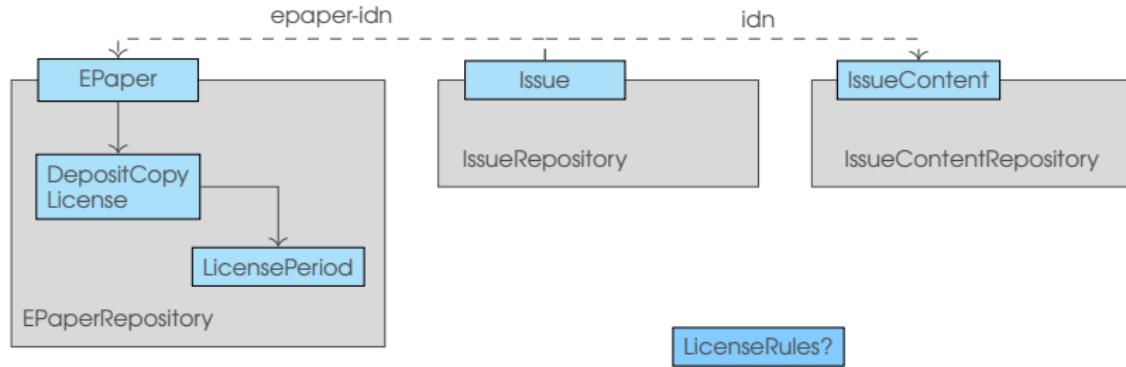
- ▶ Änderungen an dieser Schicht beeinflussen die Schicht 3 (v.a. die Entities) nicht
- ▶ Isoliert von Änderungen an der Datenbank, der graphischen Benutzeroberfläche, etc.
- ▶ Wenn sich Anforderungen ändern, hat das wahrscheinlich Auswirkungen auf diese Schicht
- ▶ Wenn sich der konkrete Betrieb der Anwendung ändert, kann das hier Auswirkungen haben

## Beispiel für Application Code

- ▶ Bankkonto-Verwaltungssoftware
- ▶ Zentraler Use Case: Überweisungen
  - ▶ Abbuchung von Konto 1, Zubuchung auf Konto 2
  - ▶ Auch hier muss eine wichtige Regel gelten
    - ⇒ Die Summe aus Abbuchung und Zubuchung ergibt immer 0
  - ▶ Kann sich ändern, beispielsweise bei Einführung von Transaktionsgebühren
    - ▶ Keine Auswirkung auf die Domäne

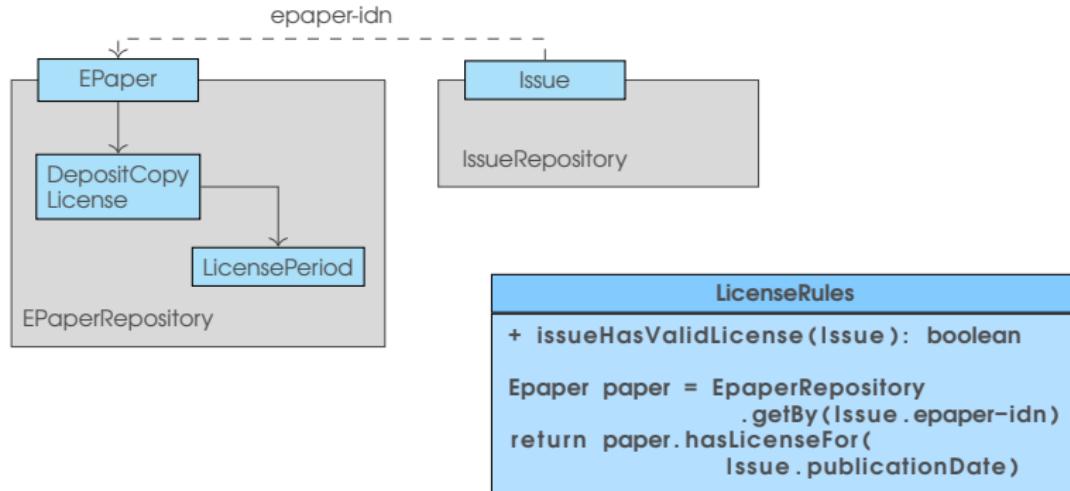
# Beispiel für DDD und Application Code

- ▶ Projekt zur Bereitstellung von E-Books
  - ▶ Prüfen, ob für eine konkrete Ausgabe eine gültige Lizenz vorliegt (LicenseRules)



# Beispiel für DDD und Application Code

- ▶ Projekt zur Bereitstellung von E-Books
  - ▶ Prüfen, ob für eine konkrete Ausgabe eine gültige Lizenz vorliegt (LicenseRules)



# Beispiel für DDD und Application Code

- ▶ Application Code (Use Cases) formuliert aggregatübergreifende Funktionalität
  - ▶ Steigt nicht in die Details eines Aggregats ab
  - ▶ Verwendet nur Methoden der Aggregate Root Entity
    - ▶ Law of Demeter light



## Schicht 1: Adapters

- ▶ Diese Schicht vermittelt Aufrufe und Daten an die inneren Schichten
  - ▶ Formatkonvertierungen
    - ▶ Externes Format wird so umgewandelt, dass die Applikation gut zurecht kommt
    - ▶ Internes Format wird so umgewandelt, dass die externen Plugins gut zurecht kommen
- ▶ Oftmals nur einfache Datenstrukturen, die hin- und hergereicht werden
- ▶ Ziel: Entkopplung von *innen* und *außen*

## Schicht 1: Adapters

- ▶ Anti-Corruption Layer
- ▶ Beispiele:
  - ▶ GUI: Enthält alle Klassen einer MVC-Struktur
  - ▶ Datenbank: Wandelt Anfragen der Anwendung in SQL-Statements um
    - ▶ Kein SQL in der Anwendung selbst!
  - ▶ GUI: Direkt verwendbares Render-Model
    - ▶ Key-Value-Paket
- ▶ Diese Schicht hält die Applikation tauglich und die Plugins frisch

## Beispiel für Adapters

- ▶ Bankkonto-Verwaltungssoftware
- ▶ Anzeige auf Webseite (HTML) vorbereiten
- ▶ Alle veränderlichen Inhalte der Seite unzweideutig berechnen (RenderModel)
  - ▶ Geldbeträge als Zeichenketten im Format 1234,56 €
    - ▶ Die Anzeigeschicht benötigt keine numerischen Werte
  - ▶ Farben als HTML-Hexcodes
  - ▶ Attribute (z.B. checked="checked" für Checkboxes)
- ▶ Ziel: Keine Umsetzungslogik in der Plugin-Schicht notwendig

# Beispiel für Adapters

- Alle Werte mundfertig im RenderModel

**FinanceStatusRenderModel  
(Map<String, String>)**

blz	94059421
user.name	Herr Max Muster
messages.new	Sie haben ...
debit	15.207,16 EUR
debit.color	#10141D
credit	-22,85 EUR
credit.color	#B9354C
overview.debit	114.497,45 EUR
...	...
...	...
accounts	List<AccountRenderModel>

**Sparkasse Musterstadt**

Unser Girokonto. Einfach mehr drin.

[Jetzt informieren](#)

dLZ: 94059421

**Finanzstatus**

Giro\*\* Haben 15.207,16 EUR  
Soll -22,85 EUR  
Saldo 15.184,96 EUR

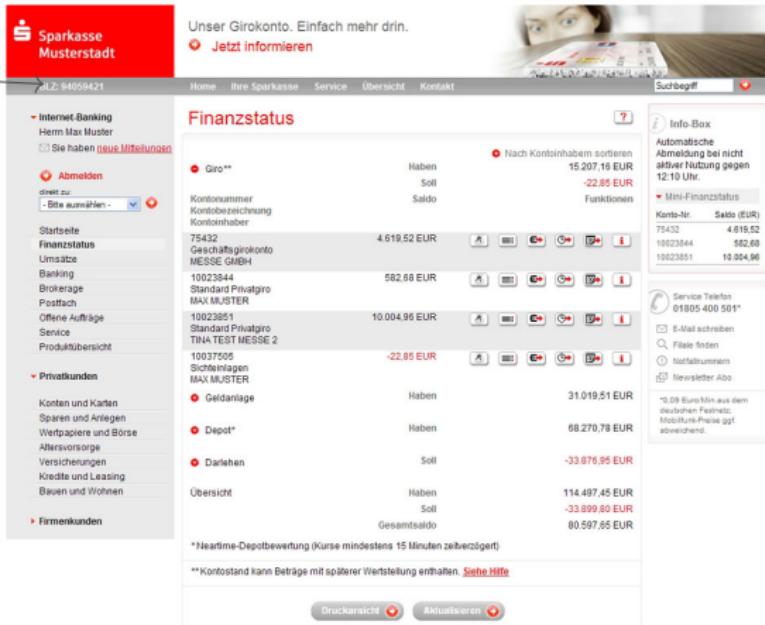
Kontonummer	Kontobezzeichnung	Kontoinhaber	Aktionen
75432	Geschäftsgirokonto	MESSE GBH	
10023844	Standard Privatgiro	MAX MUSTER	
10023851	Standard Privatgiro	TINA TEST MESSE 2	
10037505	Sichtvermögen	MAX MUSTER	
	Geldanlage		
	Depot*		
	Darlehen		

Übersicht Haben 114.497,45 EUR  
Soll -33.899,80 EUR  
Gesamtsaldo 80.597,65 EUR

\*Nachtme-Depotbewertung (Kurse mindestens 15 Minuten zeitverzögert)

\*\*Kontostand kann Beträge mit späterer Wertstellung enthalten. [Siehe Hilfe](#)

[Druckansicht](#) [Aktualisieren](#)



The screenshot shows a banking application interface. At the top, there's a red header for 'Sparkasse Musterstadt' with a logo, a call-to-action 'Jetzt informieren', and a user ID 'dLZ: 94059421'. Below the header is a green section titled 'FinanceStatusRenderModel (Map<String, String>)'. This section contains account details: blz (94059421), user.name (Herr Max Muster), messages.new (Sie haben ...), debit (15.207,16 EUR, color #10141D), credit (-22,85 EUR, color #B9354C), and overview.debit (114.497,45 EUR). There are also ellipsis rows and an 'accounts' row containing a list of 'AccountRenderModel' objects. A large arrow points from the 'accounts' row in the green section to the 'Accounts' table in the main content area. The main content area has a grey header 'Finanzstatus' with a question mark icon. It shows a summary table with columns 'Haben', 'Soll', and 'Saldo'. Below this is a table of accounts with columns 'Kontonummer', 'Kontobezzeichnung', 'Kontoinhaber', and 'Aktionen'. The table lists several accounts, including 'Giro\*\*' (Haben 15.207,16 EUR, Soll -22,85 EUR, Saldo 15.184,96 EUR), 'Standard Privatgiro' (Haben 582,68 EUR), and 'Standard Privatgiro TINA TEST MESSE 2' (Haben 10.004,96 EUR). Further down, it shows 'Sichtvermögen' (Haben 31.019,51 EUR), 'Geldanlage' (Haben 68.270,78 EUR), 'Depot\*' (Haben 0), and 'Darlehen' (Soll -33.876,95 EUR). At the bottom, there's an 'Übersicht' table with columns 'Haben', 'Soll', and 'Gesamtsaldo' (Haben 114.497,45 EUR, Soll -33.899,80 EUR, Gesamtsaldo 80.597,65 EUR). The bottom right corner features an 'Info-Box' with service information like 'Service Telefon 01805 400 501\*', 'E-Mail schreiben', 'Filete finden', 'Notfallnummern', and 'Newsletter Anmeldung'. A note at the bottom states: '\*0,09 Euro/Min aus dem deutschen Festnetz. Mobilfunk-Preise ggf. abweichen.'

DHBW Karlsruhe – vorlesung.briemla.de

33 / 69

# Beispiel für Adapters

- Alle Werte mundfertig im RenderModel

FinanceStatusRenderModel  
(Map<String, String>)

blz	94059421
user.name	Herr Max Muster
messages.new	Sie haben ...
debit	15.207,16 EUR
debit.color	#10141D
credit	-22,85 EUR
credit.color	#B9354C
overview.debit	114.497,45 EUR
...	...
...	...
accounts	List<AccountRenderModel>

Sparkasse Musterstadt

Unser Girokonto. Einfach mehr drin.

Jetzt informieren

Home Ihre Sparkasse Service Übersicht Kontakt

Finanzstatus

Giro\*\* Haben 15.207,16 EUR  
Soll -22,85 EUR  
Saldo 15.184,96 EUR

Kontonummer Kontobezzeichnung Kontoinhaber

75432 Geschäftskonto MESSE GBH	4.619,52 EUR
10023844 Standard Privatgiro MAX MUSTER	582,68 EUR
10023845 Standard Privatgiro TINA TEST MESSE 2	10.004,96 EUR
10037505 Sichtvermögen MAX MUSTER	-22,85 EUR
Geldanlage	Haben 31.019,51 EUR
Depot*	Haben 68.270,78 EUR
Darlehen	Soll -33.876,95 EUR
Übersicht	Haben 114.497,45 EUR
	Soll -33.899,80 EUR
	Gesamtsaldo 80.597,65 EUR

\*Nachtme-Depotbewertung (Kurse mindestens 15 Minuten zeitverzögert)

\*\*Kontostand kann Beträge mit späterer Wertstellung enthalten. [Siehe Hilfe](#)

Druckansicht Aktualisieren

Info-Box

Automatische Abmeldung bei nicht aktiver Nutzung gegen 12:10 Uhr.

Mini-Finanzzustand

Konto-Nr.	Saldo (EUR)
75432	4.619,52
10023844	582,68
10023845	10.004,96

Service Telefon 01805 400 501\*

E-Mail schreiben

Filete finden

Notfallnummern

Newsletter An

\*0,09 Euro/Min aus dem deutschen Festnetz. Mobilfunk-Preise ggf. abweichen.

# Beispiel für Adapters

- Alle Werte mundfertig im RenderModel

**FinanceStatusRenderModel  
(Map<String, String>)**

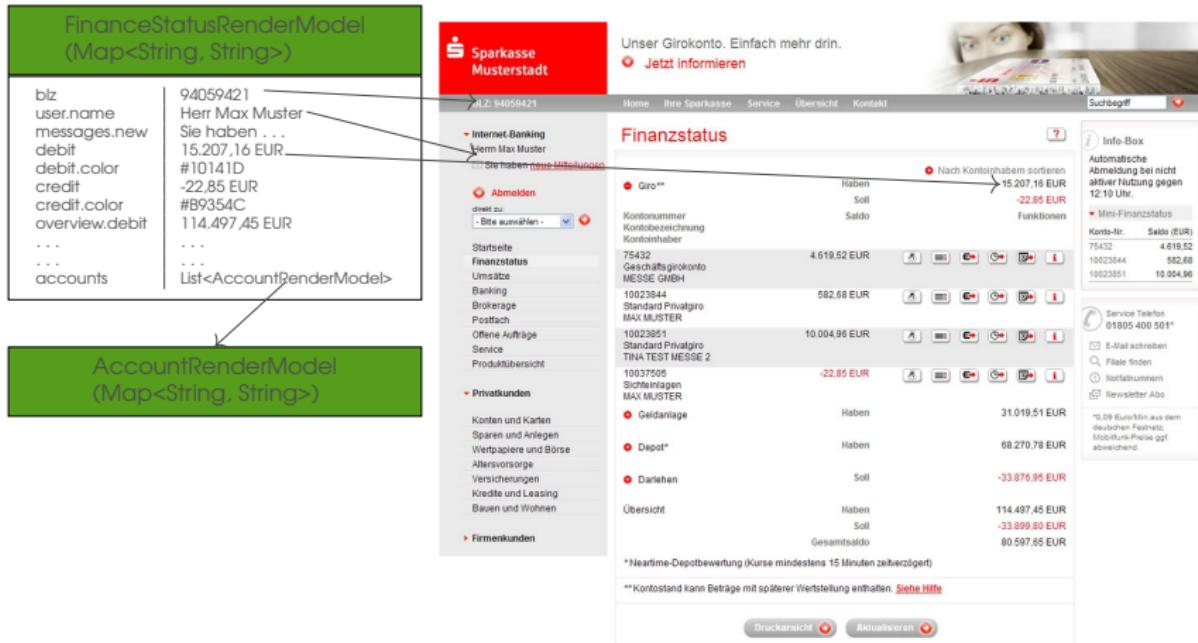
blz	94059421
user.name	Herr Max Muster
messages.new	Sie haben ...
debit	15.207,16 EUR
debit.color	#10141D
credit	-22,85 EUR
credit.color	#B9354C
overview.debit	114.497,45 EUR
...	...
...	...
accounts	List<AccountRenderModel>

The screenshot shows a banking website for 'Sparkasse Musterstadt'. At the top, there's a red header with the bank logo and a call-to-action 'Jetzt informieren'. Below it, a green banner displays the user's name 'Herr Max Muster' and a message 'Unser Girokonto. Einfach mehr drin.' A sidebar on the left lists various banking services like Internet-Banking, Finanzstatus, and Privatkunden. The main content area is titled 'Finanzstatus' and shows a table of account details. One row highlights a debit of '15.207,16 EUR' in red. Another row highlights a credit of '-22,85 EUR' in green. The table includes columns for Kontonummer, Kontobezzeichnung, Kontoinhaber, Haben, Soll, Saldo, and Funktionen. At the bottom, there are links for 'Druckansicht' and 'Aktualisieren'.

Giro**	Haben	15.207,16 EUR				
	Soll	-22,85 EUR				
	Saldo					
	Funktionen					

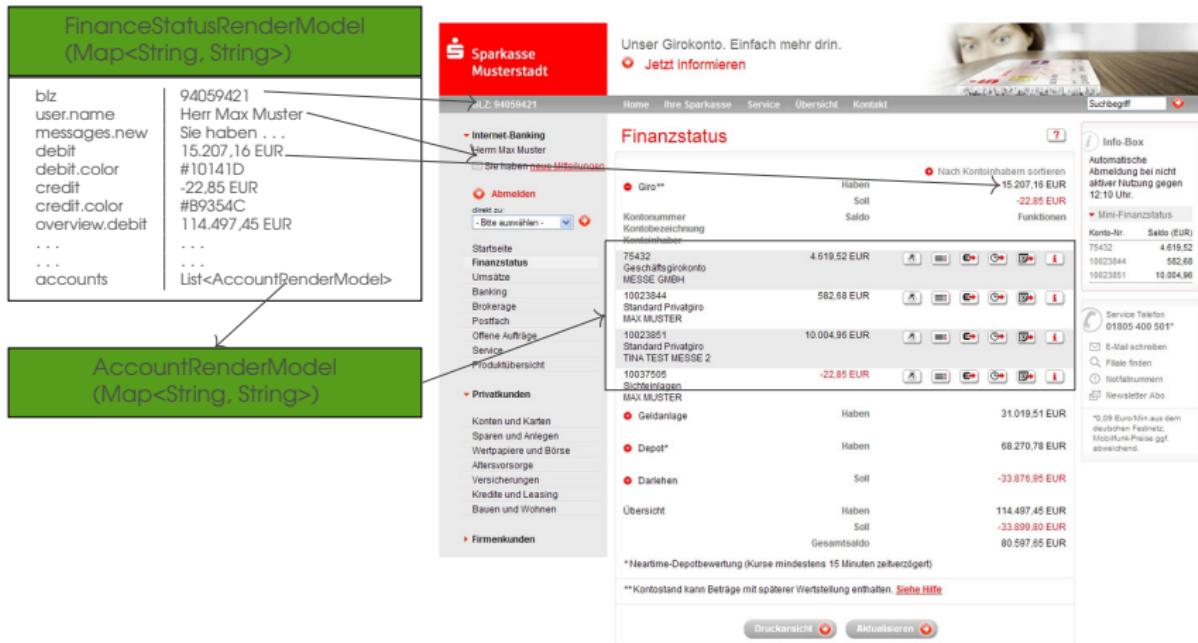
# Beispiel für Adapters

- Alle Werte mundfertig im RenderModel



# Beispiel für Adapters

- Alle Werte mundfertig im RenderModel



# Beispiel für DDD und Adapters

- ▶ Projekt zur Bereitstellung von E-Books
  - ▶ Ausgabe der Daten als REST-Webservice

Adapters	Domain Code
<pre>public class EPaperResource {      private Link selfReferringLink;     private String idn;     private String title;     private List&lt;String&gt; formerTitles;     private List&lt;String&gt; laterTitles;     private Integer yearOfFirstPublication;     private Integer yearOfLastPublication;     private List&lt;PublisherResource&gt; publishers;      @XmlJavaTypeAdapter(Link.JaxbAdapter.class)     public List&lt;Link&gt; getLinks() {         return Collections.singletonList(             selfReferringLink);     } }</pre>	<pre>public class EPaper {      private String idn;     private String title;     private final List&lt;String&gt; formerTitles;     private final List&lt;String&gt; laterTitles;     private Year yearOfFirstPublication;     private Year yearOfLastPublication;     private final List&lt;Publisher&gt; publishers; }</pre>

Implementierungsdetails  
für das Plugin  
(Serialisierung mit JAXB)

## Warum Umkopieren für Adapters?

- ▶ Warum nochmal ein Mapping von Domaindaten auf Adapterdaten?
  - ▶ Vor allem, wenn sich an den Daten nichts ändert?
- ▶ Antwort: Weil dieser Zustand temporär und zufällig ist!
  - ▶ Domain und Adapter sind momentan sehr ähnlich
  - ▶ Sie werden sich in Zukunft unabhängig voneinander verändern
  - ▶ Die Auswirkungen von Änderungen sollten möglichst lokal gehalten werden
  - ⇒ Ähnlich zu Law of Demeter

## Aber ich will trotzdem nicht!

- ▶ Es ist unnütze Arbeit ohne unmittelbaren Wert
- ▶ Das ist eine momentan korrekte Einschätzung
- ▶ Wie wäre es mit einem Kompromiss:
  - ▶ Aktuell kein Mapping einbauen
  - ▶ Sourcecode so strukturieren, dass späteres Trennen der Ebenen durch ein Mapping einfach eingebaut werden kann
  - ▶ Die Möglichkeit des Trennens immer als Werkzeug parat haben
- ▶ Arbeit dann erledigen, wenn sie einen Wert hat

## Schicht 0: Plugins

- ▶ Diese Schicht greift grundsätzlich nur auf die Adapter zu
- ▶ Enthält Frameworks, Datentransportmittel und andere Werkzeuge
  - ▶ v.a. Datenbank, Benutzeroberfläche, Web
  - ▶ Alle „Pure Fabrication“-Entscheidungen
- ▶ Wir versuchen, hier möglichst wenig Code zu schreiben
  - ▶ Hauptsächlich Delegationscode, der an die Adapter weiterleitet

## Schicht 0: Plugins

- ▶ Auf gar keinen Fall enthält diese Schicht Anwendungslogik
  - ▶ Die Daten fallen mundfertig aus dem Adapter
  - ▶ Alle Entscheidungen sind bereits gefallen
  - ▶ Anfragen werden nicht uminterpretiert (das machen die Adapter)
- ▶ Keine emotionale Bindung an diesen Code
  - ▶ Jederzeitige Änderung möglich
  - ▶ Auswirkungen nur auf die Adapterschicht
  - ▶ Übersichtlicher Aufwand

# Beispiel für Plugins

- ▶ Bankkonto-Verwaltungssoftware
- ▶ HTML-Rendering mit Velocity-Template

The screenshot shows a banking website interface. At the top left is the logo of Sparkasse Märkisch-Oderland. The top right features a red button labeled "Mehr erfahren". Below the header, there's a navigation bar with links: Home, Ihre Sparkasse, Service, Übersicht, Kontakt, and a search bar. A sidebar on the left contains links like Online-Banking, Neue Nachrichten, and Abmelden. The main content area is titled "Finanzstatus". It displays a table of account details, with the first row (Privatgirokonto) highlighted by a green border. To the right of the table is an "Info-Box" containing information about automatic logout. At the bottom right are social media icons for Facebook and Twitter.

Konto	Kontonummer Kontoname	Kontostand	Actions
Privatgirokonto ** Lebensmittel	123456	1.000,00 EUR	
Tagesgeld ** Rücklage	200905	18.235,00 EUR	
Girokonto (USD) **	654321	1.000,00 USD (734,65 EUR)	
Termingeld	223344556	15.000,00 EUR	
<b>Summe:</b>	800.100.000	47.472,05 EUR	

# Beispiel für Plugins

- ▶ Sourcecode der HTML-Seite
- ▶ Serverseitig generiert bei jedem Request

```
<tr class="tablerowodd">
    <td>Privatgirokonto <em>++</em><br>Lebensmittel<br></td>
    <td class="right" title="IBAN: DE89 1705 4040 0000 1234 56">123456<br></td>
    <td class="right"><span class="plus">1.000,00&ampnbspEUR</span><br></td>
    <td class="right">
        <input name="juhWEH" value="Kontodetails" onclick="return do();"
               src="6.gif" title="Kontodetails" type="image">
        <input name="yjSUpS" value="Umsatzabfrage" onclick="return do();"
               src="2.gif" title="Umsatzabfrage" type="image">
        <input name="ikqdyo" value="Umbuchung" onclick="return do();"
               src="3.gif" title="Umbuchung" type="image">
        <input name="cjYcZR" value="Dauerauftrag" onclick="return do();"
               src="5.gif" title="Dauerauftrag" type="image">
        <input name="gzZfjB" value="Weitere Funktionen" onclick="return do();"
               src="if5_i_aktionen.gif" title="Weitere Funktionen" type="image">
    </td>
</tr>
```

# Beispiel für Plugins

- ▶ Sourcecode der HTML-Seite
- ▶ Serverseitig generiert bei jedem Request

```
<tr class="tablerowodd">
    <td>Privatgirokonto <em>++</em><br>Lebensmittel<br></td>
    <td class="right" title="IBAN: DE89 1705 4040 0000 1234 56">123456<br></td>
    <td class="right"><span class="plus">1.000,00&ampnbspEUR</span><br></td>
    <td class="right">
        <input name="juhWEH" value="Kontodetails" onclick="return do();"
               src="6.gif" title="Kontodetails" type="image">
        <input name="yjSUpS" value="Umsatzabfrage" onclick="return do();"
               src="2.gif" title="Umsatzabfrage" type="image">
        <input name="ikqdyo" value="Umbuchung" onclick="return do();"
               src="3.gif" title="Umbuchung" type="image">
        <input name="cjYcZR" value="Dauerauftrag" onclick="return do();"
               src="5.gif" title="Dauerauftrag" type="image">
        <input name="gzZfjB" value="Weitere Funktionen" onclick="return do();"
               src="if5_i_aktionen.gif" title="Weitere Funktionen" type="image">
    </td>
</tr>
```

# Beispiel für Plugins

- ▶ Sourcecode der HTML-Seite
- ▶ Serverseitig generiert bei jedem Request

```
<tr class="tablerowodd">
    <td>Privatgirokonto <em>++</em><br>Lebensmittel<br></td>
    <td class="right" title="IBAN: DE89 1705 4040 0000 1234 56">123456<br></td>
    <td class="right"><span class="plus">1.000,00&ampnbspEUR</span><br></td>
    <td class="right">
        <input name="juhWEH" value="Kontodetails" onclick="return do();"
               src="6.gif" title="Kontodetails" type="image">
        <input name="yjSUpS" value="Umsatzabfrage" onclick="return do();"
               src="2.gif" title="Umsatzabfrage" type="image">
        <input name="ikqdyo" value="Umbuchung" onclick="return do();"
               src="3.gif" title="Umbuchung" type="image">
        <input name="cjYcZR" value="Dauerauftrag" onclick="return do();"
               src="5.gif" title="Dauerauftrag" type="image">
        <input name="gzZfjB" value="Weitere Funktionen" onclick="return do();"
               src="if5_i_aktionen.gif" title="Weitere Funktionen" type="image">
    </td>
</tr>
```

# Beispiel für Plugins

- ▶ Sourcecode der HTML-Seite
- ▶ Serverseitig generiert bei jedem Request

```
<tr class="tablerowodd">
    <td>Privatgirokonto <em>++</em><br>Lebensmittel<br></td>
    <td class="right" title="IBAN: DE89 1705 4040 0000 1234 56">123456<br></td>
    <td class="right"><span class="plus">1.000,00&ampnbspEUR</span><br></td>
    <td class="right">
        <input name="juhWEH" value="Kontodetails" onclick="return do();"
               src="6.gif" title="Kontodetails" type="image">
        <input name="yjSUpS" value="Umsatzabfrage" onclick="return do();"
               src="2.gif" title="Umsatzabfrage" type="image">
        <input name="ikqdyo" value="Umbuchung" onclick="return do();"
               src="3.gif" title="Umbuchung" type="image">
        <input name="cjYcZR" value="Dauerauftrag" onclick="return do();"
               src="5.gif" title="Dauerauftrag" type="image">
        <input name="gzZfjB" value="Weitere Funktionen" onclick="return do();"
               src="if5_i_aktionen.gif" title="Weitere Funktionen" type="image">
    </td>
</tr>
```

# Beispiel für Plugins

- ▶ Sourcecode der HTML-Seite
- ▶ Serverseitig generiert bei jedem Request

```
<tr class="tablerowodd">
    <td>Privatgirokonto <em>+</em><br>Lebensmittel<br></td>
    <td class="right" title="IBAN: DE89 1705 4040 0000 1234 56">123456<br></td>
    <td class="right"><span class='plus'>1.000,00&ampnbspEUR</span><br></td>
    <td class="right">
        <input name="juhWEH" value="Kontodetails" onclick="return do();"
               src="6.gif" title="Kontodetails" type="image">
        <input name="yjSUpS" value="Umsatzabfrage" onclick="return do();"
               src="2.gif" title="Umsatzabfrage" type="image">
        <input name="ikqdyo" value="Umbuchung" onclick="return do();"
               src="3.gif" title="Umbuchung" type="image">
        <input name="cjYcZR" value="Dauerauftrag" onclick="return do();"
               src="5.gif" title="Dauerauftrag" type="image">
        <input name="gzZfjB" value="Weitere Funktionen" onclick="return do();"
               src="if5_i_aktionen.gif" title="Weitere Funktionen" type="image">
    </td>
</tr>
```

# Beispiel für Plugins

- ▶ Sourcecode der HTML-Seite
- ▶ Serverseitig generiert bei jedem Request

```
<tr class="tablerowodd">
    <td>Privatgirokonto <em>++</em><br>Lebensmittel<br></td>
    <td class="right" title="IBAN: DE89 1705 4040 0000 1234 56">123456<br></td>
    <td class="right"><span class='plus'>1.000,00&ampnbspEUR</span><br></td>
    <td class="right">
        <input name="juhWEH" value="Kontodetails" onclick="return do();"
               src="6.gif" title="Kontodetails" type="image">
        <input name="yjSUpS" value="Umsatzabfrage" onclick="return do();"
               src="2.gif" title="Umsatzabfrage" type="image">
        <input name="ikqdyo" value="Umbuchung" onclick="return do();"
               src="3.gif" title="Umbuchung" type="image">
        <input name="cjYcZR" value="Dauerauftrag" onclick="return do();"
               src="5.gif" title="Dauerauftrag" type="image">
        <input name="gzZfjB" value="Weitere Funktionen" onclick="return do();"
               src="if5_i_aktionen.gif" title="Weitere Funktionen" type="image">
    </td>
</tr>
```

# Beispiel für Plugins

- ▶ Veränderliche Inhalte als benannte Variablen
- ▶ Velocity setzt die Werte des RenderModel ein

```
<tr class="tablerowodd">
    <td>$account_title<br></td>
    <td class="right" title="">$iban">$number </td>
    <td class="right"><span class="$sgn">$balance</span><br></td>
    <td class="right">
```

AccountRenderModel (Map<String, String>)	
account_title	Privatgirokonto<em>**</em> Lebensmittel 
iban	IBAN: DE89 1705 4040 0000 1234 56
number	123456
sgn	plus
balance	1.000,00 EUR

# Beispiel für Plugins

- Die entstandene Webseite enthält keinen Hinweis auf Variablen oder das Rendering

The screenshot shows the homepage of the Sparkasse Märkisch-Oderland Online-Banking. The top navigation bar includes links for Home, Ihre Sparkasse, Service, Übersicht, Kontakt, and a search bar. A sidebar on the left lists various banking services like Online-Banking, Finanzstatus, Kontodetails, Termingeld, Umsätze, Banking, PIN/TAN-Verwaltung, Brokerage, Deka, and Kreditkarte. The main content area displays the 'Finanzstatus' section, which shows a table of accounts. One account, 'Privatgirokonto \*\* Lebensmittel', is highlighted with a green border. To the right of the table is an 'Info-Box' containing information about automatic logout. The bottom right corner features social media icons for Facebook and Twitter.

Konto	Kontonummer Kontoname	Kontostand	Actions
Privatgirokonto ** Lebensmittel	123456	1.000,00 EUR	
Tagesgeld ** Rücklage	200905	18.235,00 EUR	
Girokonto (USD) **	654321	1.000,00 USD (734,65 EUR)	
Termingeld	223344556	15.000,00 EUR	

# Beispiel für DDD und Plugins

- ▶ Projekt zur Bereitstellung von E-Books
  - ▶ Anbindung ans Internet per JAX-RS

```
@Path("/epapers")
public class EPapersEndpoint {
    @Inject private EPaperAccess ePaperAccess;
    @Inject private EPaperToEPaperResourceMapper ePaperToEPaperResource;

    @GET
    @Path("")
    @Produces(MediaType.APPLICATION_JSON + "; charset=utf-8")
    public Response findEPapers(
        @QueryParam("iIn")
        @NotNull(message = "add required query parameter iIn") String holdingsIIn) {
        return Response.ok(
            ePaperAccess.findEPapers(holdingsIIn).stream()
                .map(ePaperToEPaperResource)
                .collect(Collectors.toList()))
            .build();
    }
}
```

# Beispiel für DDD und Plugins

- ▶ Projekt zur Bereitstellung von E-Books
  - ▶ Anbindung ans Internet per JAX-RS

```
@Path("/epapers")
public class EPapersEndpoint {
    @Inject private EPaperAccess ePaperAccess;
    @Inject private EPaperToEPaperResourceMapper ePaperToEPaperResource;

    @GET
    @Path("")
    @Produces(MediaType.APPLICATION_JSON + "; charset=utf-8")
    public Response findEPapers(
        @QueryParam("iIn")
        @NotNull(message = "add required query parameter iIn") String holdingsIn) {
        return Response.ok(
            ePaperAccess.findEPapers(holdingsIn).stream()
                .map(ePaperToEPaperResource)
                .collect(Collectors.toList()))
            .build();
    }
}
```

Application Code

# Beispiel für DDD und Plugins

- ▶ Projekt zur Bereitstellung von E-Books
  - ▶ Anbindung ans Internet per JAX-RS

```
@Path("/epapers")
public class EPapersEndpoint {
    @Inject private EPaperAccess ePaperAccess;
    @Inject private EPaperToEPaperResourceMapper ePaperToEPaperResource;

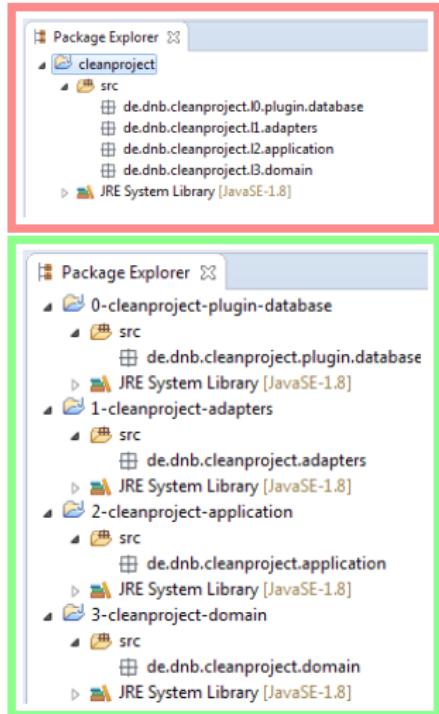
    @GET
    @Path("")
    @Produces(MediaType.APPLICATION_JSON + "; charset=utf-8")
    public Response findEPapers(
        @QueryParam("iIn")
        @NotNull(message = "add required query parameter iIn") String holdingsIIn) {
        return Response.ok(
            ePaperAccess.findEPapers(holdingsIIn).stream()
                .map(ePaperToEPaperResource)
                .collect(Collectors.toList()))
            .build();
    }
}
```

Application Code

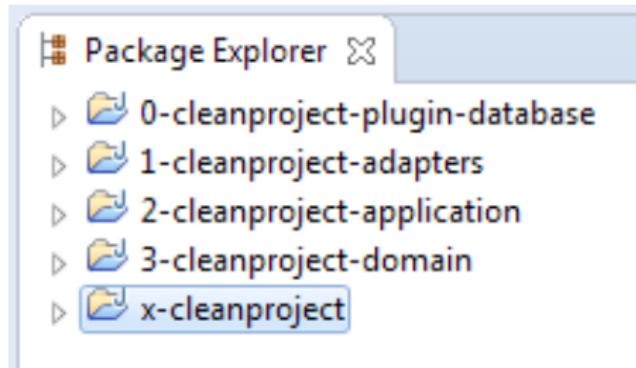
Adapters

# Konkrete Umsetzung

- ▶ Nicht alle Klassen in einem Projekt
  - ▶ Schichtenbildung über Packages ist in Ordnung
  - ▶ Aber: keine Überprüfung durch den Compiler
- ▶ Lieber mehrere Projekte („Multi-Projekt“)
  - ▶ Compiler findet nur Klassen
    - ▶ im eigenen Projekt
    - ▶ in referenzierten Projekten



# Konkrete Umsetzung: Maven



```
<project>
    <groupId>de.dnb</groupId>
    <artifactId>x-cleanproject</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>pom</packaging>

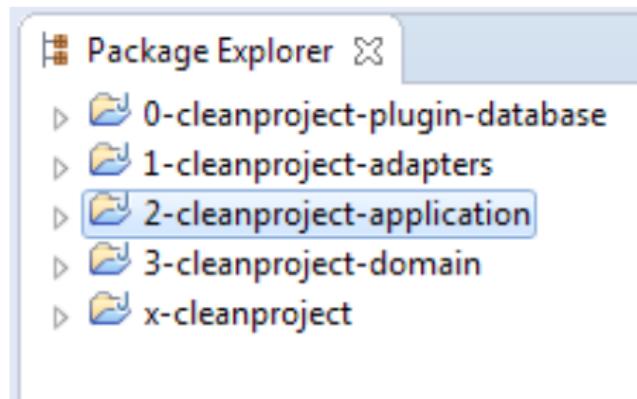
    <modules>
        <module>3-cleanproject-domain</module>
        <module>2-cleanproject-application</module>
        <module>1-cleanproject-adapters</module>
        <module>0-cleanproject-plugin-database</module>
    </modules>

    (...)

</project>
```

- ▶ Ein „Klammerprojekt“ für globale Einstellungen
  - ▶ Maven „Parent-POM“
- ▶ Enthält alle anderen Projekte als Module

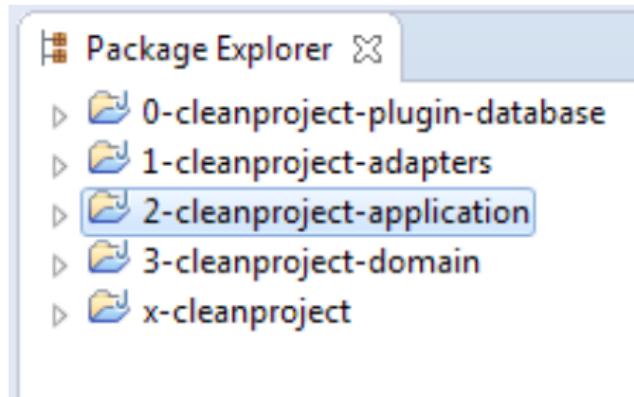
# Konkrete Umsetzung: Maven



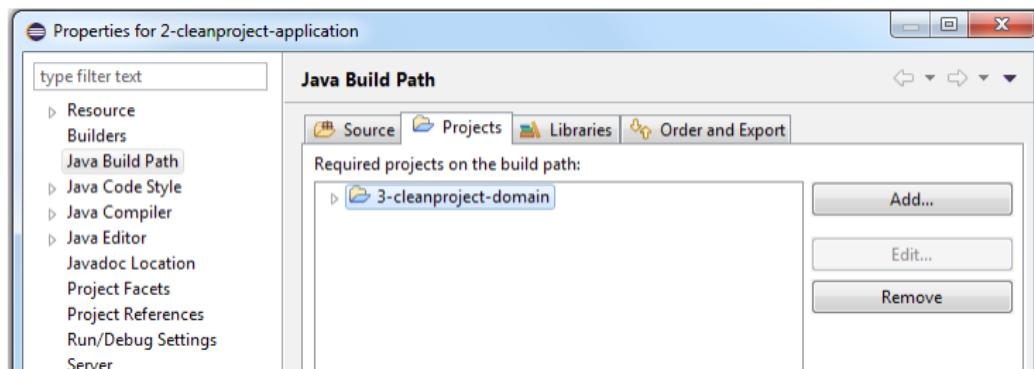
```
<project>
    <artifactId>2-cleanproject-application</artifactId>
    <dependencies>
        <dependency>
            <artifactId>3-cleanproject-domain</artifactId>
            <groupId>de.dnb</groupId>
            <scope>compile</scope>
        </dependency>
    </dependencies>
    <parent>
        <artifactId>x-cleanproject</artifactId>
        <groupId>de.dnb</groupId>
    </parent>
    (...)
```

- ▶ Projekt 2 soll von Projekt 3 abhängen
  - ▶ Im pom.xml als Dependency eintragen

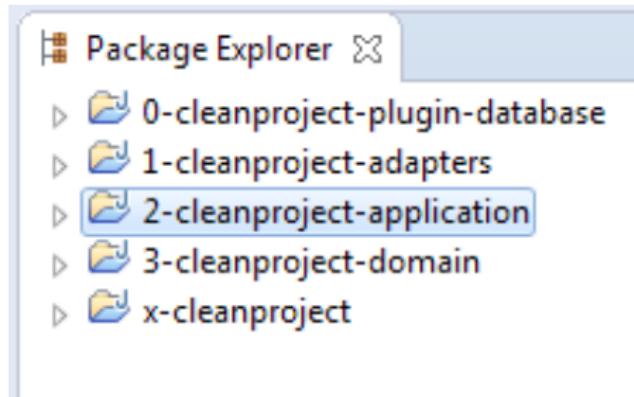
# Konkrete Umsetzung: Manuell



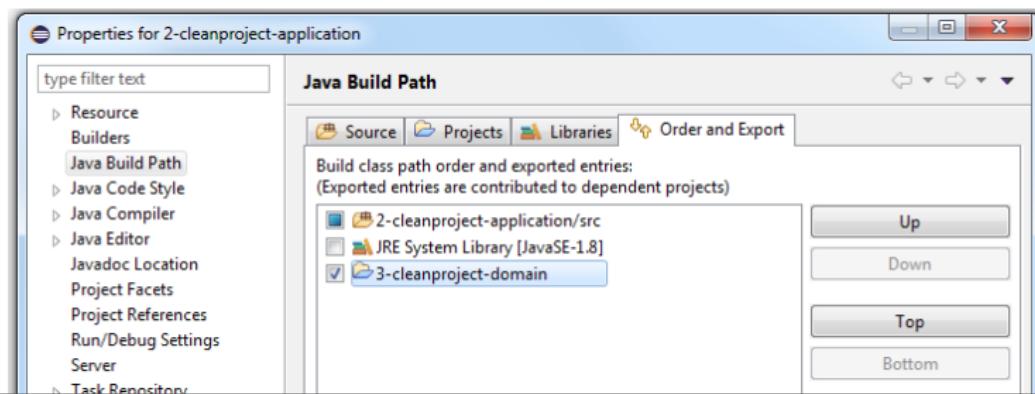
- ▶ Projekt 2 soll von Projekt 3 abhängen
  - ▶ In den Eclipse-Projekteinstellungen angeben



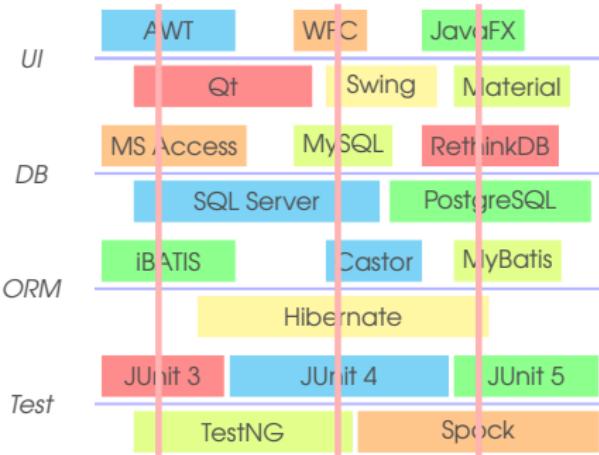
# Konkrete Umsetzung: Manuell



- ▶ Transitive Abhängigkeiten freigeben
  - ▶ In den Eclipse-Projekteinstellungen



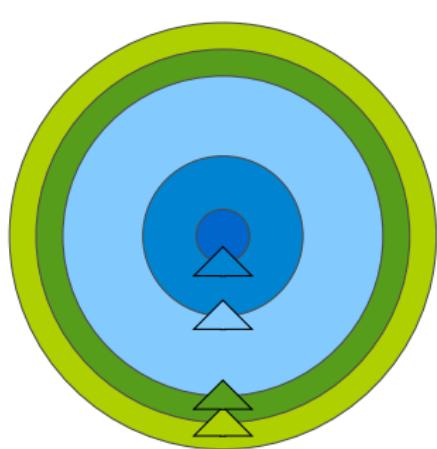
# Review der Technologiewahl



- ▶ Auswahl der Technologie zu Beginn eines Projekts
  - ▶ Hat starken Einfluß auf die Entwicklung
- ⇒ Altert und veraltet zusammen mit der Anwendung

	Tag 1	Tag 2	Tag 3
AWT		Swing	JavaFX
MS Access	MS Access	MySQL	RethinkDB
iBATIS		Hibernate	MyBatis
JUnit		TestNG	Spock

# Ziel der Clean Architecture

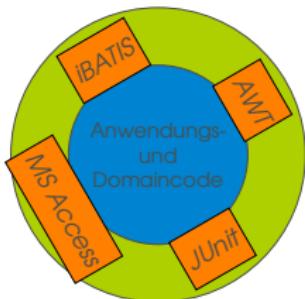


- ▶ Das Ziel der Clean Architecture ist, Code nur von langlebigerem Code abhängig zu machen
- ▶ Wenn sich Technologien ändern müssen, kann die Anwendung unverändert bleiben

# Grenzen der Clean Architecture

- ▶ Technische Grundlagen müssen stabil\* bleiben
  - ▶ Plattform SDK (bei Java das JDK)
  - ▶ Programmiersprache (bei Java die Java-Syntax)
  - ▶ Compiler (bei Java der javac)
  - ▶ Laufzeitumgebung (bei Java die JVM)
- ▶ Auch Betriebssystem und Hardware benötigen ausreichende Stabilität
- ▶ Das ist ein Grund, warum immer noch Cobol auf Mainframes produktiv betrieben wird

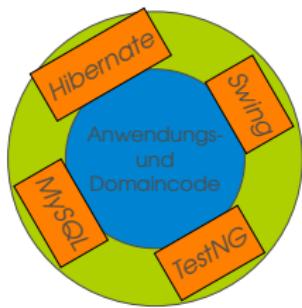
# Clean Architecture Technologiewahl



- ▶ Anwendung von Technologiewahl nicht betroffen
  - ▶ Konkrete Technologien sind nur noch Plugins
    - ▶ „Details“
- ⇒ Können einzeln ersetzt werden

Tag 1	Tag 2	Tag 3
AWT	Swing	JavaFX
MS Access	MySQL	RethinkDB
iBATIS	Hibernate	MyBatis
JUnit	TestNG	Spock

# Clean Architecture Technologiewahl

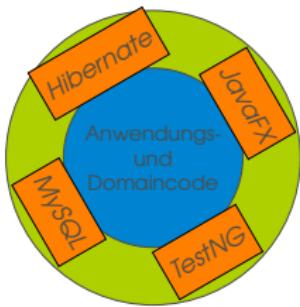


- ▶ Ersetzen einer Technologie ändert die Anwendung nicht
- ▶ Adapter müssen wahrscheinlich angepasst werden

⇒ Alle Anforderungen bleiben erhalten

Tag 1	Tag 2	Tag 3
AWT	Swing	JavaFX
MS Access	MySQL	RethinkDB
iBATIS	Hibernate	MyBatis
JUnit	TestNG	Spock

# Clean Architecture Technologiewahl



- ▶ Jedes Plugin kann einzeln ersetzt werden
  - ▶ Keine oder nur minimale Abhängigkeiten zwischen Plugins
- ⇒ Separation of Concerns

Tag 1	Tag 2	Tag 3
AWT	Swing	JavaFX
MS Access	MySQL	RethinkDB
iBATIS	Hibernate	MyBatis
JUnit	TestNG	Spock

# Positionierung: Beispiel 1

Use Case

```
public class ChangeUserPassword {  
    private final AuthenticationService authenticationService;  
  
    @Inject  
    public ChangeUserPassword(  
        final AuthenticationService authenticationService)  
    {  
        this.authenticationService = authenticationService;  
    }  
  
    @Transactional  
    public boolean changeUserPassword(  
        @NotNull final Authentication authentication,  
        @NotNull final String oldPassword,  
        @NotNull final String newPassword) {  
        final boolean oldPasswordIsValid =  
            authenticationService.checkPassword(  
                authentication,  
                oldPassword);  
        if (oldPasswordIsValid) {  
            authenticationService.setUserPassword(  
                authentication.getLoginName(),  
                newPassword);  
        }  
        return oldPasswordIsValid;  
    }  
}
```

Plugins

Adapters

Application Code

Domain Code

# Positionierung: Beispiel 2

Entity

```
public class User {  
  
    @NotNull private String loginName;  
    @NotNull private String fullName;  
    @NotNull private String emailAddress;  
  
    protected User() {  
    }  
  
    public static UserBuilder create() {  
        return new UserBuilder();  
    }  
  
    public String getLoginName() {  
        return loginName;  
    }  
  
    public String getFullName() {  
        return fullName;  
    }  
  
    public String getEmailAddress() {  
        return emailAddress;  
    }  
    (...)  
}
```

Plugins

Adapters

Application Code

Domain Code

# Positionierung: Beispiel 3

Framework Driver

Plugins

Adapters

Application Code

Domain Code

```
public class JPAAuthenticationService implements AuthenticationService {  
    private JPAUserEntityRepository userEntityRepository;  
    private JPAAuthenticationEntityRepository authenticationRepository;  
  
    @Inject  
    public JPAAuthenticationService(  
        final JPAUserEntityRepository userEntityRepository,  
        final JPAAuthenticationEntityRepository repository)  
    {  
        this.userEntityRepository = userEntityRepository;  
        this.authenticationEntityRepository = repository;  
    }  
  
    @Override  
    public boolean checkPassword(  
        @NotNull final Authentication authentication,  
        @NotNull final String password) {  
        return authenticationRepository  
            .findAuthenticationEntityByLoginNameAndPassword(  
                authentication.getLoginName(),  
                password)  
            .isPresent();  
    }  
    (...)  
}
```

# Clean Architecture und Frameworks



- ▶ Frameworks streben oft die Alleinherrschaft an
- ▶ Abhängigkeiten zeigen oft vom Anwendungscode in das Framework
  - ⇒ Das ist die falsche Richtung
- ⇒ Abhängigkeiten immer von außen nach innen

# Clean Architecture und Frameworks



- ▶ Frameworks streben oft die Alleinherrschaft an
- ▶ Abhängigkeiten zeigen oft vom Anwendungscode in das Framework
  - ⇒ Das ist die falsche Richtung
  - ⇒ Abhängigkeiten immer von außen nach innen

# Clean Architecture und Frameworks



- ▶ Frameworks streben oft die Alleinherrschaft an
- ▶ Abhängigkeiten zeigen oft vom Anwendungscode in das Framework
  - ⇒ Das ist die falsche Richtung
- ⇒ Abhängigkeiten immer von außen nach innen

# Clean Architecture und Frameworks



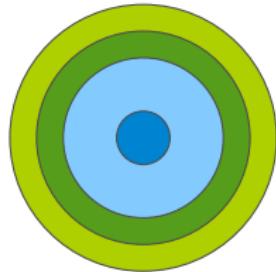
- ▶ Frameworks streben oft die Alleinherrschaft an
- ▶ Abhängigkeiten zeigen oft vom Anwendungscode in das Framework
  - ⇒ Das ist die falsche Richtung
  - ⇒ Abhängigkeiten immer von außen nach innen

# Clean Architecture und Frameworks



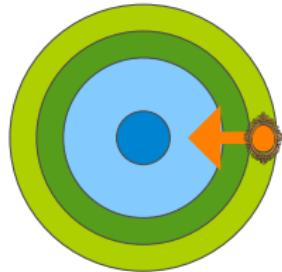
- ▶ Frameworks streben oft die Alleinherrschaft an
- ▶ Abhängigkeiten zeigen oft vom Anwendungscode in das Framework
  - ⇒ Das ist die falsche Richtung
  - ⇒ Abhängigkeiten immer von außen nach innen

## Frameworks positionieren



- ▶ Frameworks sind Details
- ▶ Details sind Plugins
- ▶ Plugins gehören *an den Rand* der Anwendung
- ▶ Das Framework ausfüllen heißt, Aufrufe an die Anwendung zu delegieren
- ▶ Problematisch bei Frameworks mit Metaprogrammierung

## Frameworks positionieren



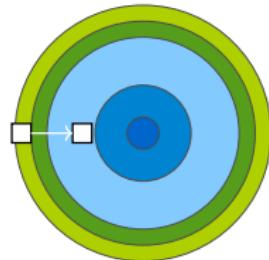
- ▶ Frameworks sind Details
- ▶ Details sind Plugins
- ▶ Plugins gehören *an den Rand* der Anwendung
- ▶ Das Framework ausfüllen heißt, Aufrufe an die Anwendung zu delegieren
- ▶ Problematisch bei Frameworks mit Metaprogrammierung

## Frameworks separieren

- ▶ Am besten den Code inkl. Framework in eigenes Projekt auslagern
  - ▶ Framework-Projekt referenziert Anwendungs-Projekt
- ▶ Anwendung muss unabhängig vom Framework bau- und betreibbar sein
- ▶ Die Schnittstelle für das Framework-Projekt wird eventuell sehr spezifisch ausfallen
  - ▶ Versuchung widerstehen, eine universelle Schnittstelle zu entwickeln
  - ▶ *Throwaway-Adapter*, d.h. Code, der verzichtbar ist

# Übergabe von Daten

- ▶ An einer Schichtgrenze müssen Daten übergeben werden
- ▶ Von außen nach innen ist einfach
  - ▶ Parameter eines Methodenaufrufs

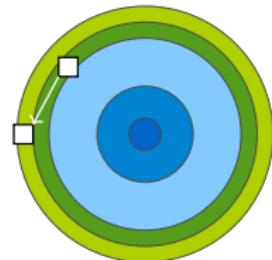


```
@Path( "/epapers" )
public class EPapersEndpoint {
    @GET
    @Path( "" )
    @Produces(MediaType.APPLICATION_JSON + "; charset=utf-8")
    public Response findEPapers(
        @QueryParam("iIn")
        @NotNull(message = "add required query parameter iIn") String holdingsIIn) {
        return Response.ok(
            ePaperAccess.findEPapers(holdingsIIn).stream()
                .map(ePaperToEPaperResource)
                .collect(Collectors.toList())
                .build());
    }
}
```

The code snippet shows Java code for a REST endpoint. It defines a class `EPapersEndpoint` with a single method `findEPapers`. The method takes a query parameter `iIn` and returns a `Response` object. The response is generated by calling `ePaperAccess.findEPapers(holdingsIIn)`, which is annotated with `@NotNull` and `@Produces`. The resulting stream is mapped to `EPaperResource` objects and collected into a list before being built into a response. A callout box labeled "Application Code" points to the line `ePaperAccess.findEPapers(holdingsIIn)`.

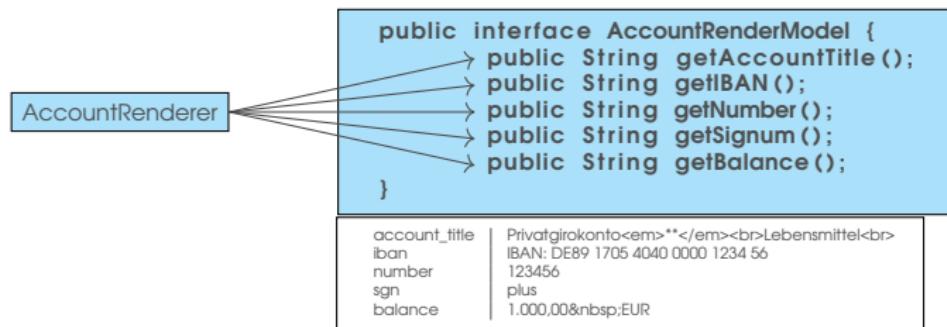
# Übergabe von Daten

- ▶ Von innen nach außen gibt es zwei grundsätzliche Möglichkeiten
  - ▶ Reaktiv: Als Rückgabewert eines Methodenaufrufs von außen
  - ▶ Aktiv: Rückruf (Callback) einer äußeren Methode von innen
- ▶ Sicht von außen:
  - ▶ Reaktiv == Pull (Außen muss „ziehen“)
  - ▶ Aktiv == Push (Innen meldet sich von alleine)



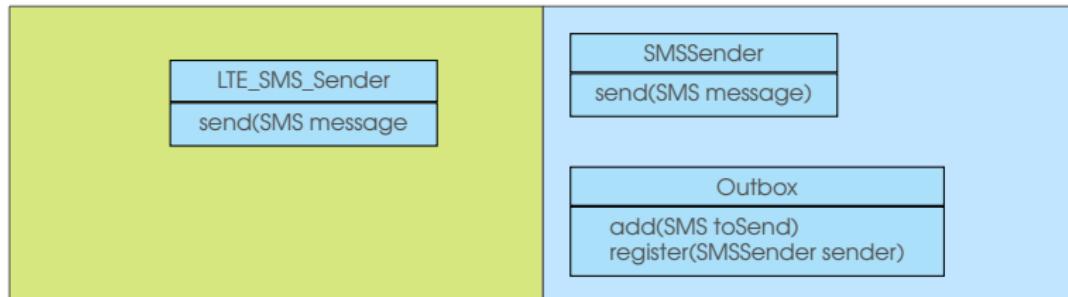
# Daten reaktiv herausgeben

- ▶ Das äußere Plugin fragt zu einem ihm genehmen Zeitpunkt nach den Daten
- ▶ Die inneren Schichten antworten nur



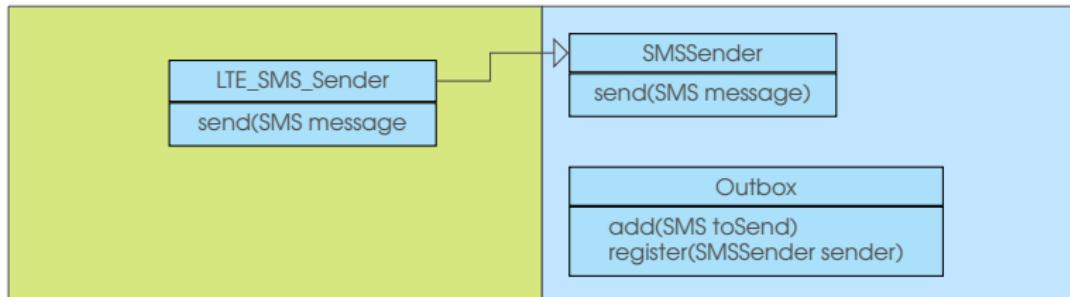
## Daten aktiv herausgeben

- ▶ Das äußere Plugin meldet sich zu einem frühen Zeitpunkt als Befehlsempfänger an
- ▶ Die inneren Schichten geben die Befehle zum ihnen genehmten Zeitpunkt
- ▶ Oft als Beobachter-(Listener)-Muster realisiert



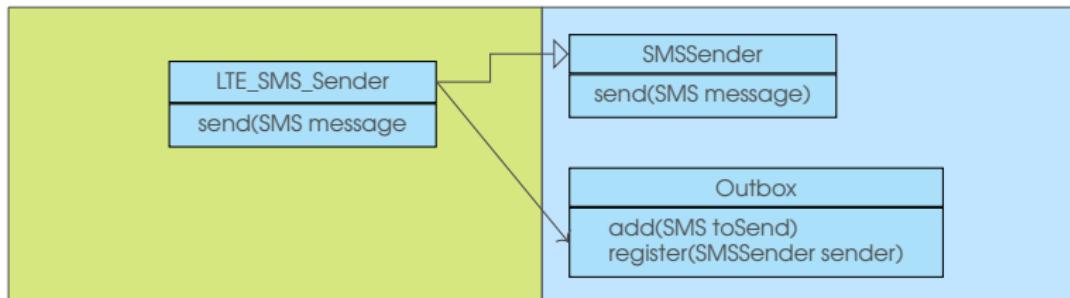
## Daten aktiv herausgeben

- ▶ Das äußere Plugin meldet sich zu einem frühen Zeitpunkt als Befehlsempfänger an
- ▶ Die inneren Schichten geben die Befehle zum ihnen genehmten Zeitpunkt
- ▶ Oft als Beobachter-(Listener)-Muster realisiert



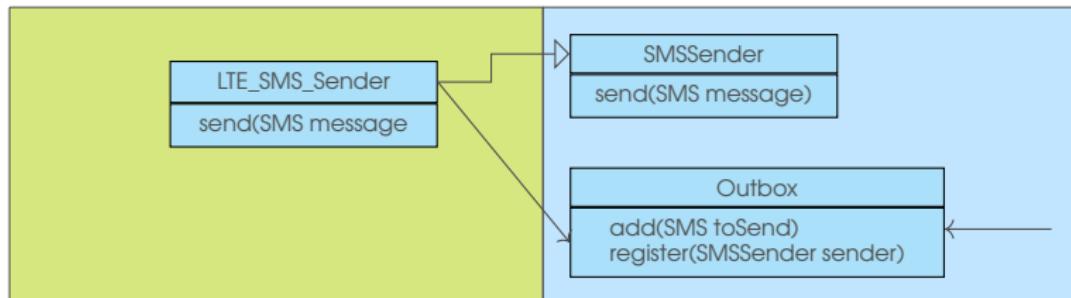
## Daten aktiv herausgeben

- ▶ Das äußere Plugin meldet sich zu einem frühen Zeitpunkt als Befehlsempfänger an
- ▶ Die inneren Schichten geben die Befehle zum ihnen genehmten Zeitpunkt
- ▶ Oft als Beobachter-(Listener)-Muster realisiert



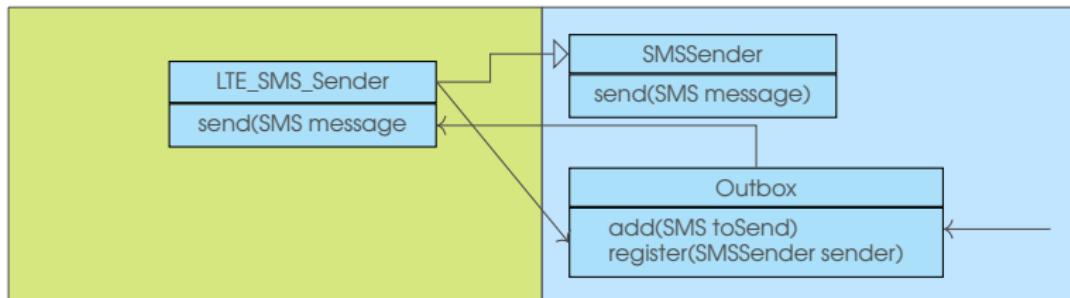
## Daten aktiv herausgeben

- ▶ Das äußere Plugin meldet sich zu einem frühen Zeitpunkt als Befehlsempfänger an
- ▶ Die inneren Schichten geben die Befehle zum ihnen genehmten Zeitpunkt
- ▶ Oft als Beobachter-(Listener)-Muster realisiert



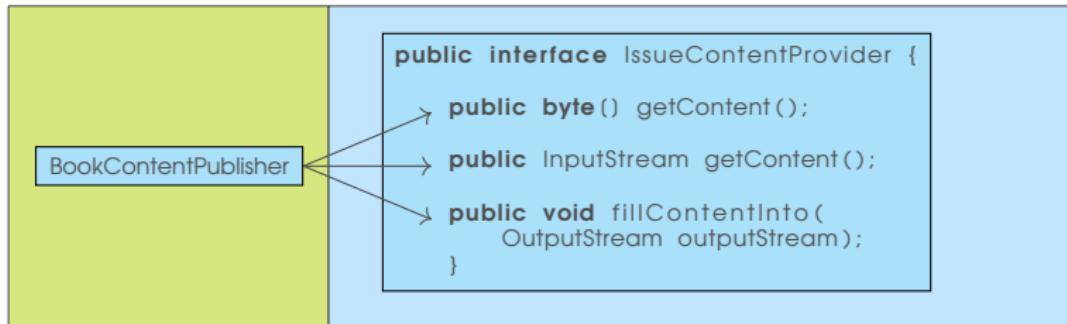
# Daten aktiv herausgeben

- ▶ Das äußere Plugin meldet sich zu einem frühen Zeitpunkt als Befehlsempfänger an
- ▶ Die inneren Schichten geben die Befehle zum ihnen genehmten Zeitpunkt
- ▶ Oft als Beobachter-(Listener)-Muster realisiert

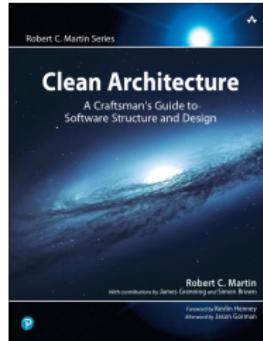


## Daten aktiv herausgeben

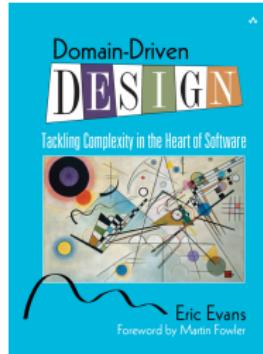
- ▶ Variante: Die Datenverarbeitungsrichtung umdrehen
  - ▶ OutputStream geben statt InputStream geben lassen
- ▶ Beispielsweise bei Bereitstellung von E-Books



# Literatur

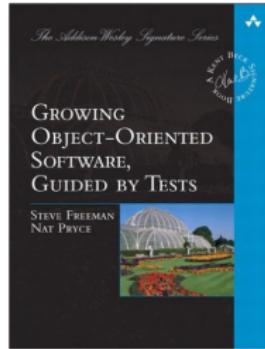


- ▶ Clean Architecture
  - ▶ Robert C. Martin
  - ▶ Pearson Education
  - ▶ ISBN: 978-0134494166



- ▶ Domain-Driven Design
  - ▶ Eric Evans
  - ▶ Addison-Wesley
  - ▶ ISBN: 978-0321125217

# Literatur



## ► Growing Object-Oriented Software, Guided by Tests

- Steve Freeman und Nat Pryce
- Addison-Wesley
- ISBN: 978-0321503626

# Literatur

- ▶ Hexagonal Architecture
  - ▶ <http://alistair.cockburn.us/Hexagonal+architecture>
- ▶ The Onion Architecture
  - ▶ <http://jeffreypalermo.com/blog/the-onion-architecture-part-1>
- ▶ The Clean Architecture
  - ▶ <https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html>
- ▶ Layers, Onions, Ports, Adapters: it's all the same
  - ▶ <http://blog.ploeh.dk/2013/12/03/layers-onions-ports-adapters-its-all-the-same>

## Weitere Quellen

- ▶ Monolith
  - ▶ By Source, Fair use,  
<https://en.wikipedia.org/w/index.php?curid=31738209>
- ▶ Oval Baroque Gold Frame
  - ▶ Fotolia Datei #77261068 | Urheber: dmitrygolikov
- ▶ Ausmalbuch
  - ▶ <http://www.traum-salon.de/pages/buecher/uebersicht/herr-wolke-lese-raetsel-ausmalbuch.php>
- ▶ Trend für Stressabbau - Ausmalbuch für Erwachsene
  - ▶ Fotolia Datei: #102219361 | Urheber: moltaprop
- ▶ Bricklayer worker installing brick masonry on exterior wall
  - ▶ Fotolia Datei: #117356924 | Urheber: Hoda Bogdan
- ▶ Suspicious Looking Device
  - ▶ Junkfunnel Labs (Casey Smith) <http://art.junkfunnel.com/?p=83>