Model Driven Architecture and Model Driven Design at Ing-DiBa

Markus Demetz / Claudiu Cherloaba

Wien, den 16. Januar 2017

ING DiBa

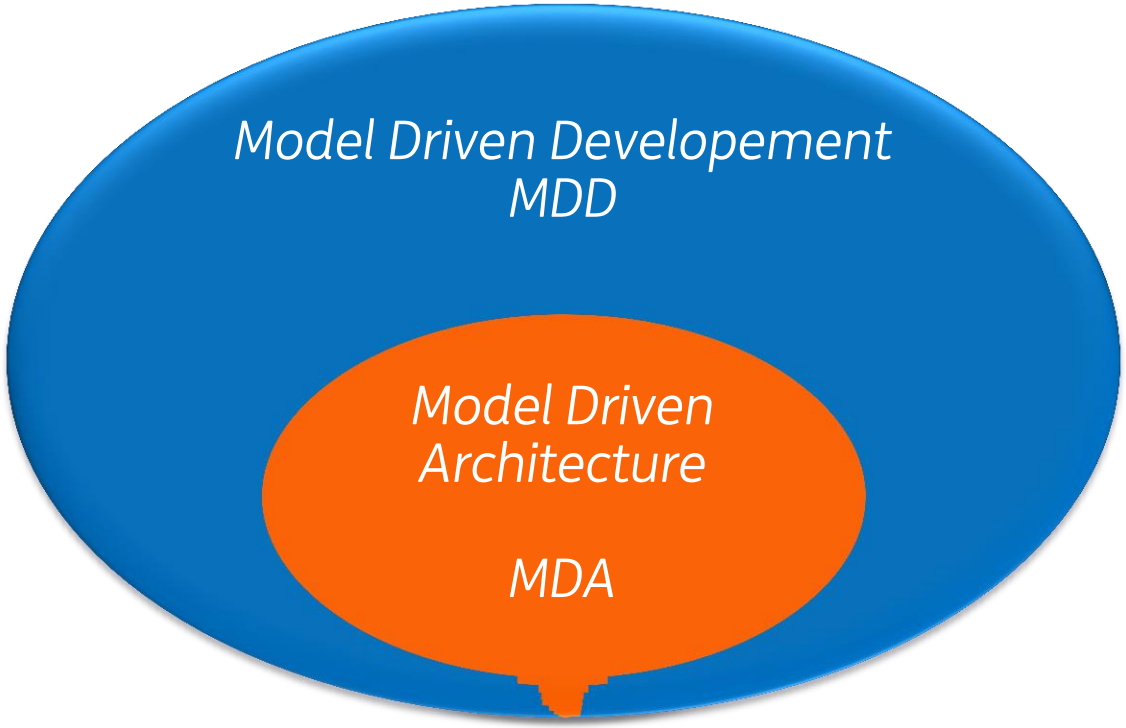
Was ist Model Driven Architecture?

Model Driven Architecture (MDA) ist ein spezieller Ansatz von Model Driven Developement (MDD).

MDA basiert auf Standards, welche von der Object Management Group (OMG) festgelegt werden.

Bessere Handhabung der Komplexität von Großen Systemen und klare Trennung der Abstraktionsebenen.

Modelle stehen im Mittelpunkt des Paradigmas



ING DiBa

Was ist ein Modell und welche Arten werden unterschieden?

Ein MDA-Modell ist eine abstrakte Repräsentation von *Struktur*, *Funktion* oder *Verhalten* eines Systems. Die Spezifikation einer Komponente ist unabhängig von der technischen Umsetzung zu beschreiben. Man unterscheidet:

- **Computation Independent Model (CIM)**
  Beschreibt ein Softwaresystem auf fachlicher Ebene und definiert, was das System leistet. Ist der Modelltyp mit der höchsten Abstraktionsebene. Dient als Kommunikation zwischen technischen und fachlichen Experten.

- **Platform Independent Models (PIM)**
  Beschreibt Funktionalität, Struktur und Verhalten, indem es von der Implementierung des Systems abstrahiert.

- **Platform Specific Models (PSM)**
  Beschreibt ein Modell, welches an einer speziellen Plattform (z. B. J2EE / JEE) gebunden ist. Das PSM erweitert das PIM um Aspekte die rein plattform-spezifisch sind bzw. um die konkreten technischen Details.

Modellierungssprache

Üblicherweise wird mit Hilfe von UML modelliert.

Im Prinzip setzt das MDA-Paradigma auf beliebigen Modellierungssprachen auf, welche durch eine Meta-Modellierungssprache definiert werden.

Meta Object Facility (MOF) legt generische Konstrukte zur Modellierung fest (Meta-MetaModell).

Eine Modellierungssprache ist eine Instanz des Meta-MetaModells und ist ein MetaModell.

Ein Modell ist eine Instanz des MetaModells.

UML hat sich in der Praxis erfolgreich durchgesetzt und kann mit Hilfe von UML-Profilen projektspezifisch angepasst werden.

Rational Software Architekt (RSA) von IBM

Warum Model Driven Architecture?

Abbildung des Softwareerstellungsprozesses durch Modelle

Ein hoher Anteil der Software kann generativ, d. h. durch Transformation der Modelle erzeugt werden.

Fachliche und technische Aspekte werden durch Plattform unabhängige Modelle getrennt.

Plattform = eine Menge von Technologien, welche Schnittstellen zur Verfügung stellen.

Plattform Model (PM) definiert technische Konzepte und stellt Elemente für die Modellierung eines plattform-spezifischen Software Systems zur Verfügung.

Die Repräsentation von Software wird von der Programmcodeebene auf die Modellebene verlegt.

Aus dem PSM kann Programmcode automatisch generiert werden.

Ziele von Model Driven Architecture

- **Steigerung der Entwicklungsgeschwindigkeit**
  Automation durch Formalisierung
  Codegenerierung auf Basis formal eindeutiger Modelle -> Generative Programmierung

- **Bessere Handhabbarkeit aufgrund höherer Abstraktion**
  Programmierung auf abstrakter Ebene (klare Trennung von fachlichen und technischen Anteilen)
  Technologieunabhängige Beschreibung von zentralen Konzepten ermöglicht eine leichtere Anpassung beim Technologiewandel.
  Platform Independet Models überleben einen Technologiewandel.

- **Erstellung einer Softwarearchitektur in Form eines UML-Diagramms.**
  Code kann automatisch erzeugt und „von Hand“ vervollständigt werden.

- **Ein Auseinanderlaufen von Modell und Code wird unterbunden**

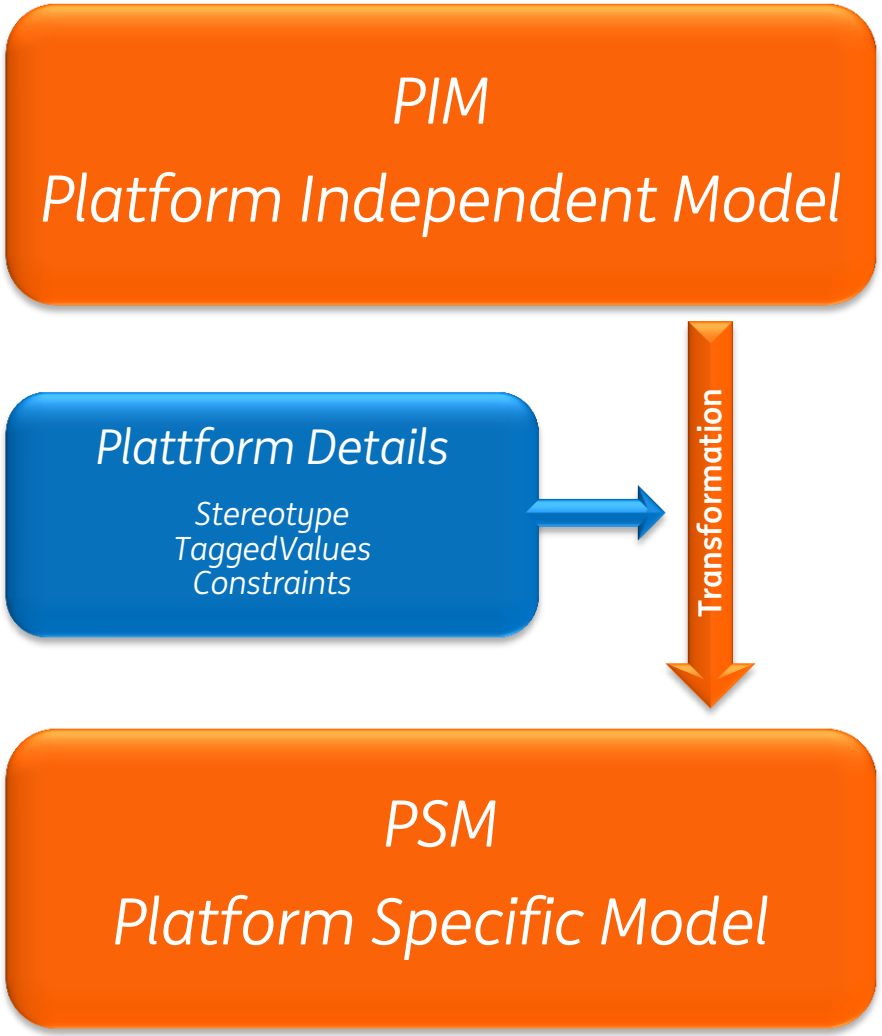
Transformation von Modellen

Modelle werden durch die Auswahl von Plattformen in weniger abstrakte Modelle transformiert bis letztendlich ausführbarer Programmcode entsteht.

Das PIM wird nur einmal modelliert.
Das Mapping auf die von der Architektur zur Verfügung gestellten Platform wird mit Hilfe von Tools durchgeführt.

Der größte Nutzen von MDA ist die Generierung von Programmcode.

Oft erfolgt die Generierung des Codes gleich aus dem PIM, ohne die Zwischenstufe eines PSMs



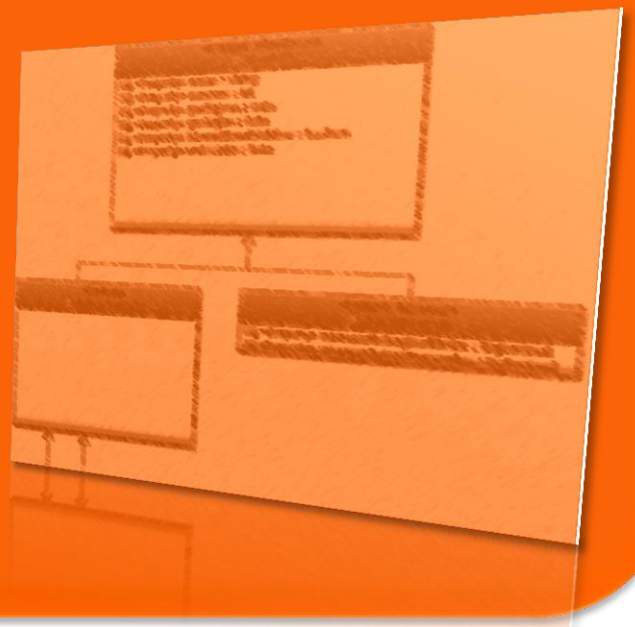
ING DiBa

Was wird modelliert?

Modellierung der Struktur

Domainmodell

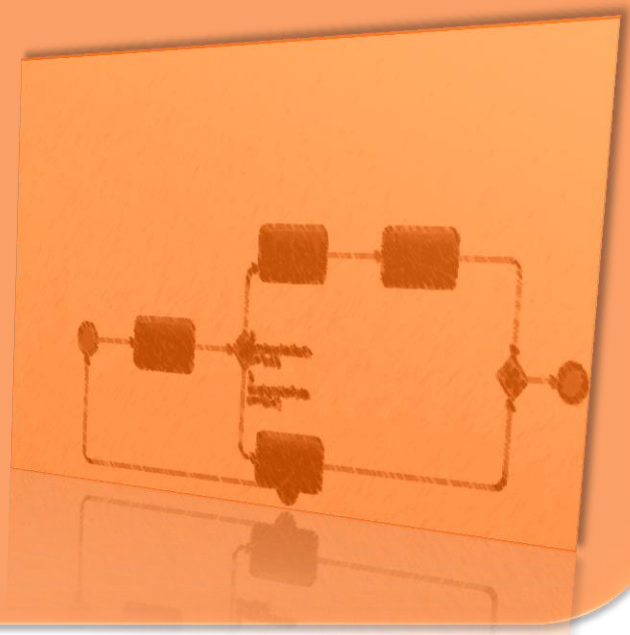
UML-KLassendiagramm



Modellierung des Verhaltens

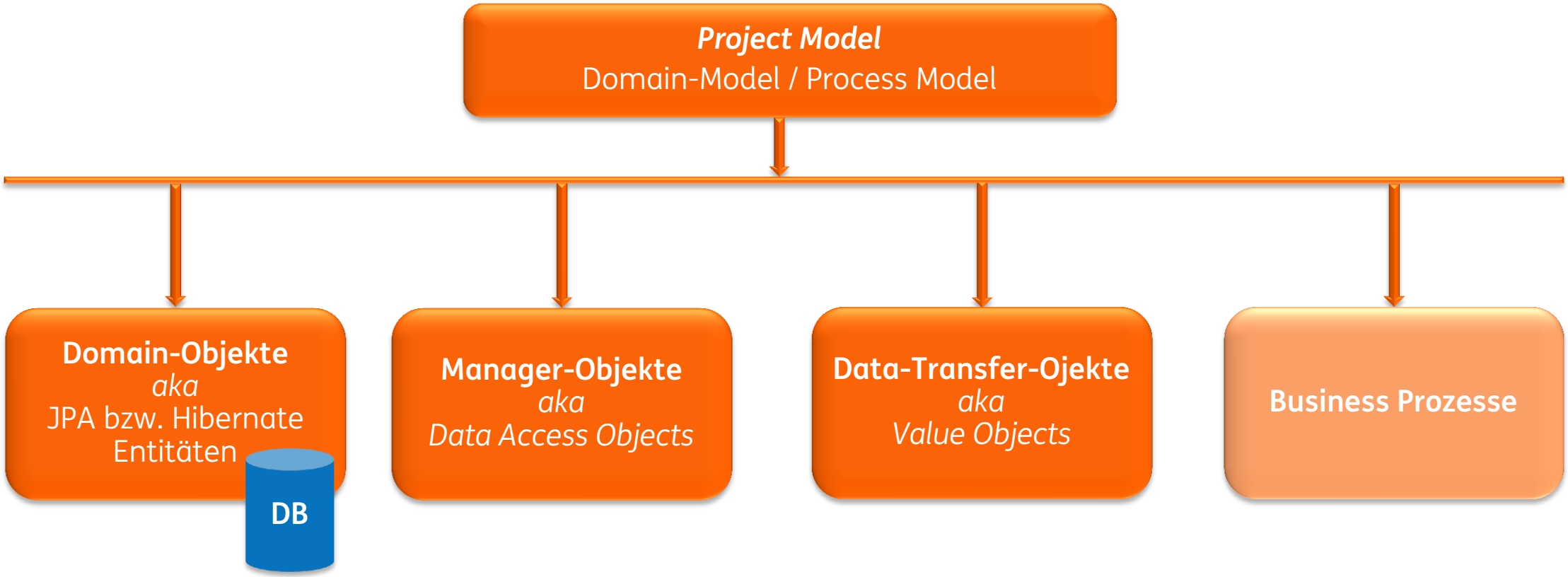
Prozessmodell

Zustandsdiagramm



ING DiBa

Was wird generiert?



JPA annotierte Java-Classen für das *Object-Relational-Mapping* auf die Datenbank (**ORM**)

Java-Objekte mit der Zugriffslogik auf die Datenbank. Verwaltet die Domain-Objekte.

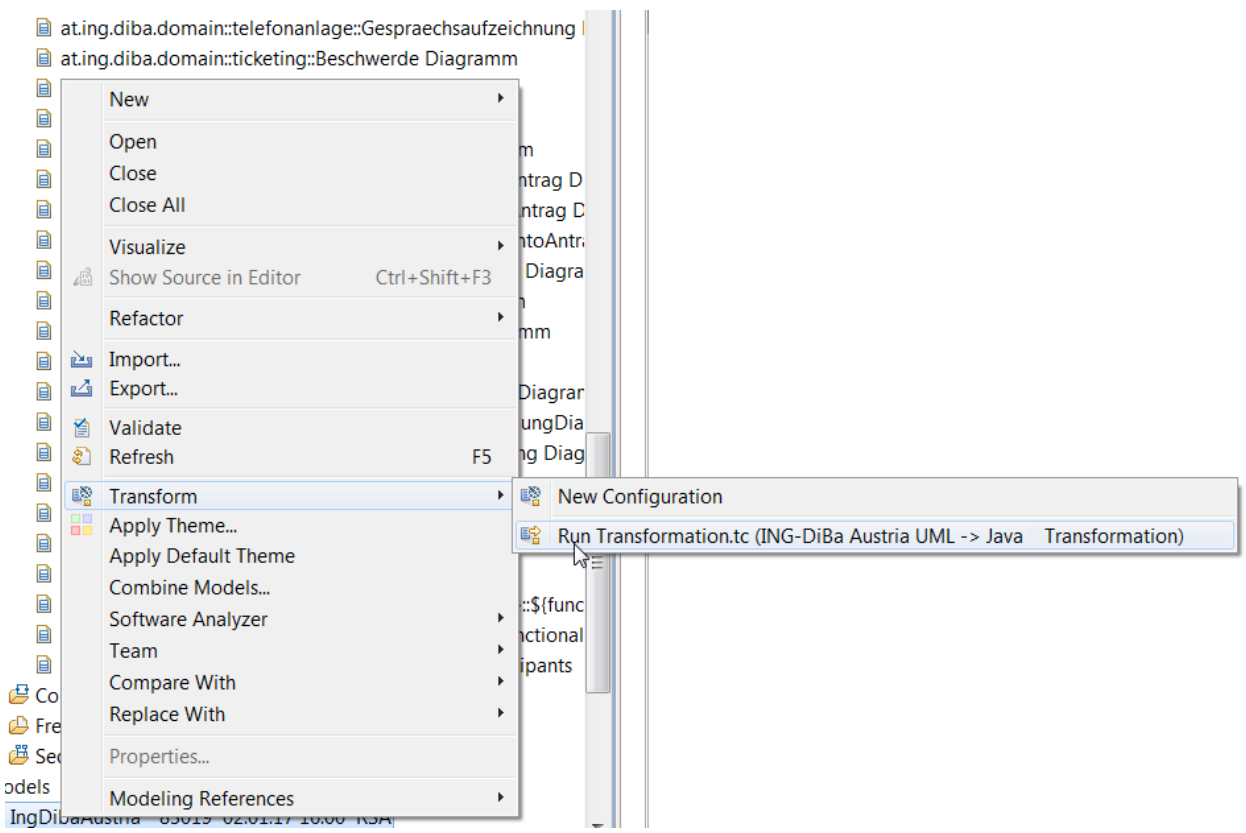
Java-Objekte (POJOs) für den Transfer von Domain-Daten über Transaktionsgrenzen hinaus.

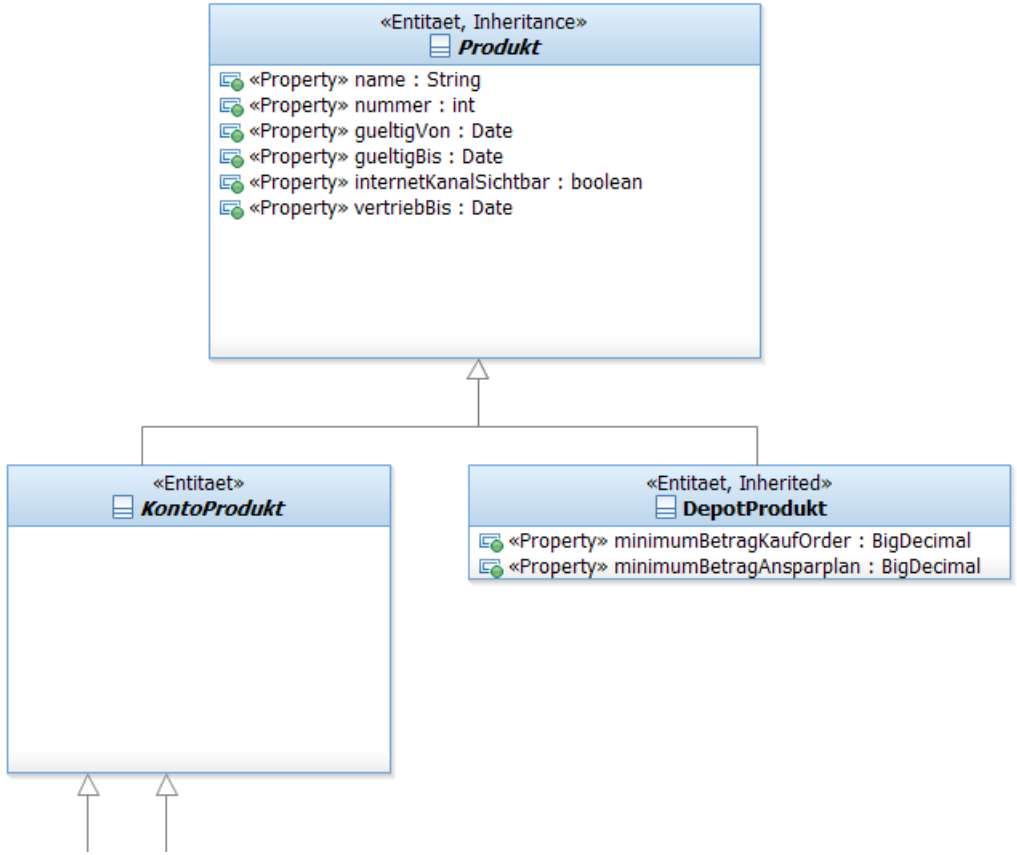
Java-Klassen für die Abarbeitung von Business-Prozessen. Klassen müssen händisch implementiert werden.

ING DiBa

Generierung von Programmcode

Das selbstentwickelte Plug-In kann über das Kontext-Menü aufgerufen werden generiert Java-Code



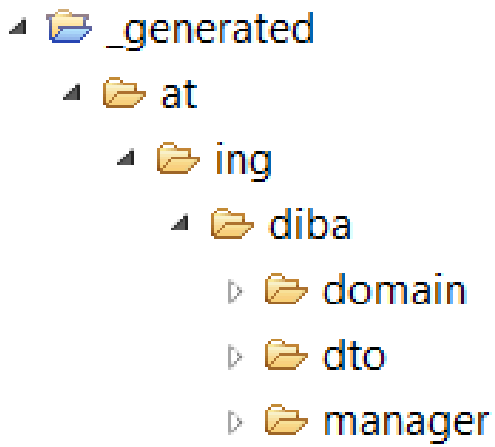


ING DiBa

Programmcode

```
public class #Classname {

#for(attribute in attributes)
  #renderAttribute(attribute)
#endfor

#for(attribute in attributes)
  #renderAccessMethod(attribute)
#endfor

}
```


```
public class Produkt {

  private String name;

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }
}
```



Der generierte Programmcode wird in die Projekt-Struktur kopiert.

ING DiBa

Vorteile der Code-Generierung

- **Qualität**
  Durch den templatebasierten Ansatz können Bugs im Template behoben und somit flächendeckend beseitigt werden.

- **Konsistenz**
  Klassen-, Methoden- und Argumentennamen werden einheitlich vergeben und sind somit über den gesamten generierten Code konsistent

- **Produktivität**
  Die Generierung ist schneller als handgeschriebener Code. Der wahre Benefit macht sich erst bei erneuter Code-Generierung bemerkbar, wenn der Code an neue Anforderungen und Bedürfnisse angepasst werden muss.

- **Abstraktion**
  Das Design bleibt abstrakt und frei von Implementierungsdetails, sodass der Generator an neue Technologien und technischen Platformen angepasst werden kann.

Anforderungen an Model Driven Architecture

Minimierung der Qualistätsrisiken

Maximierung der Automation

Nach der Transformation nur geringfügige bzw. keine Ergänzungen nötig

Vorteile / Nachteile von Model Driven Architecture

Vorteile

- Bessere Übersicht über das Design
- Langlebigkeit der Modelle
- Nachvollziehbarkeit des Systems
- Änderungen am System können leichter durchgeführt werden
- Leichtere Fehlerkorrektur

Nachteile

- Modelle sind nicht leicht unter den Tools austauschbar
  Tools verwenden meistens proprietäre Formate zur Speicherung von Modellen und bieten Import/Export-Schnittstellen für XMI an, welche oft nicht vollständig implementiert sind.

- Tools sind oftmals sehr komplex
- Großer Anfangsaufwand

Process Engine

- Wir benützen eine In-House-Entwicklung
- In-House-Entwicklung löste jBPM ab. (Business Process Management)
  - JBPM konnte zur damaligen Zeit den Anforderungen nicht gerecht werden.
  - Wenig skalierbar und hatte Bugs die nicht ausgebessert wurden.
  - Die Möglichkeit für Postkorb-Einträge war nicht gegeben.

Vorteile der Verwendung von Prozessen

- Visuelle Grundlage der einzelnen Prozessschritte
- Logging der einzelnen Prozessschritte
- Leichtere Kommunikation mit Fachabteilung und Business Analyse durch die grafische Darstellung.

- TODO

Beispiel: Modellierung von Entitäten

Anforderung:

Kontonummer *
Bitte geben Sie eine gültige Kontonummer ein.

213412342134

Nachname *

Grausam

Vorname

Gregor

Prämie für den Werber *

50 Euro CliniClowns Spendenprämie

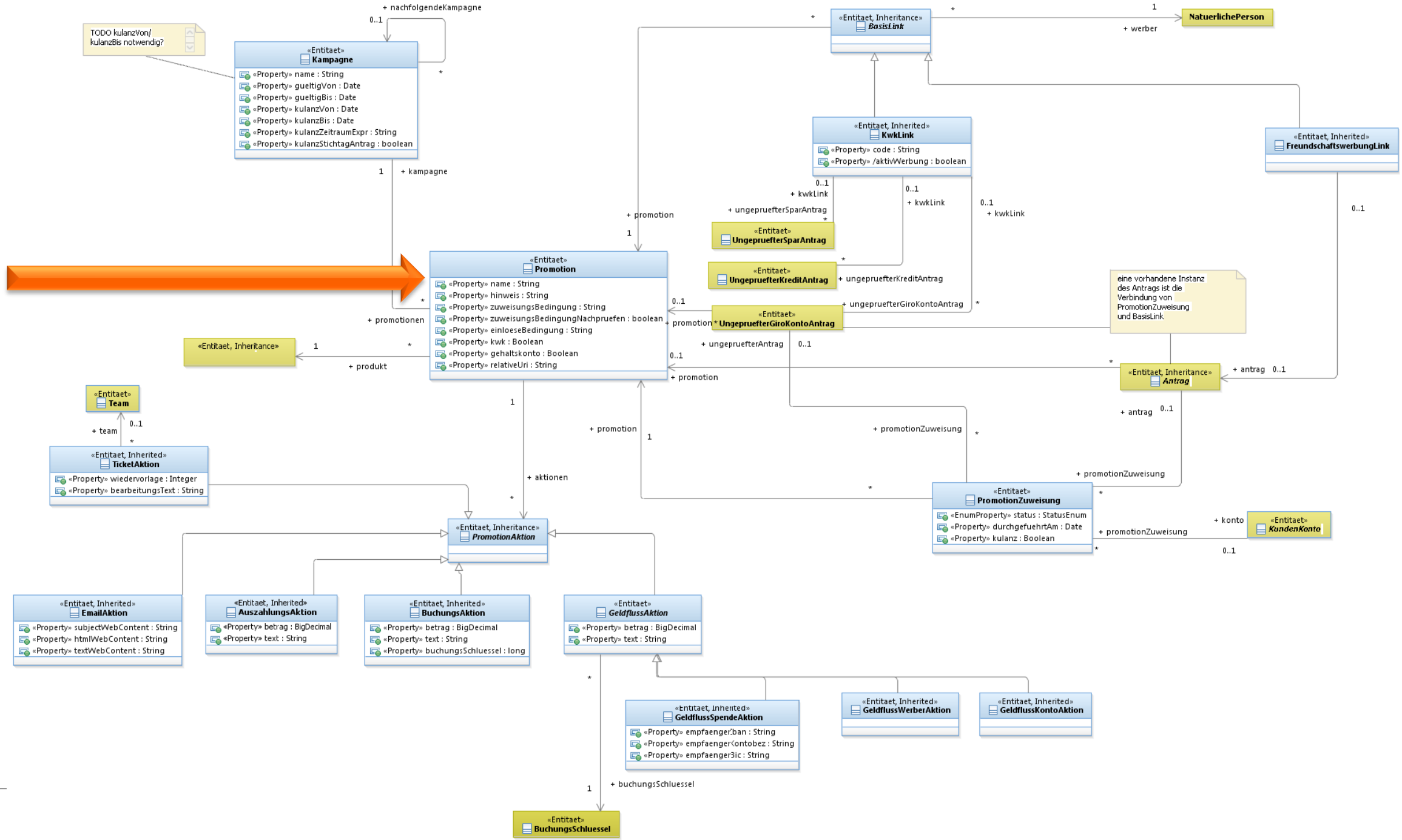
Hinweis:

Wenn sich der Werber für die „CliniClowns-Spende“ entscheidet, erklärt er sich damit einverstanden, dass die ING-DiBa die auf sein Direkt-Sparkonto gebuchte Spendenprämie an die CliniClowns weiterüberweist und seine personenbezogenen Daten zum ausschließlichen Zweck der Erstellung einer Spendenbestätigung an die CliniClowns weitergibt.

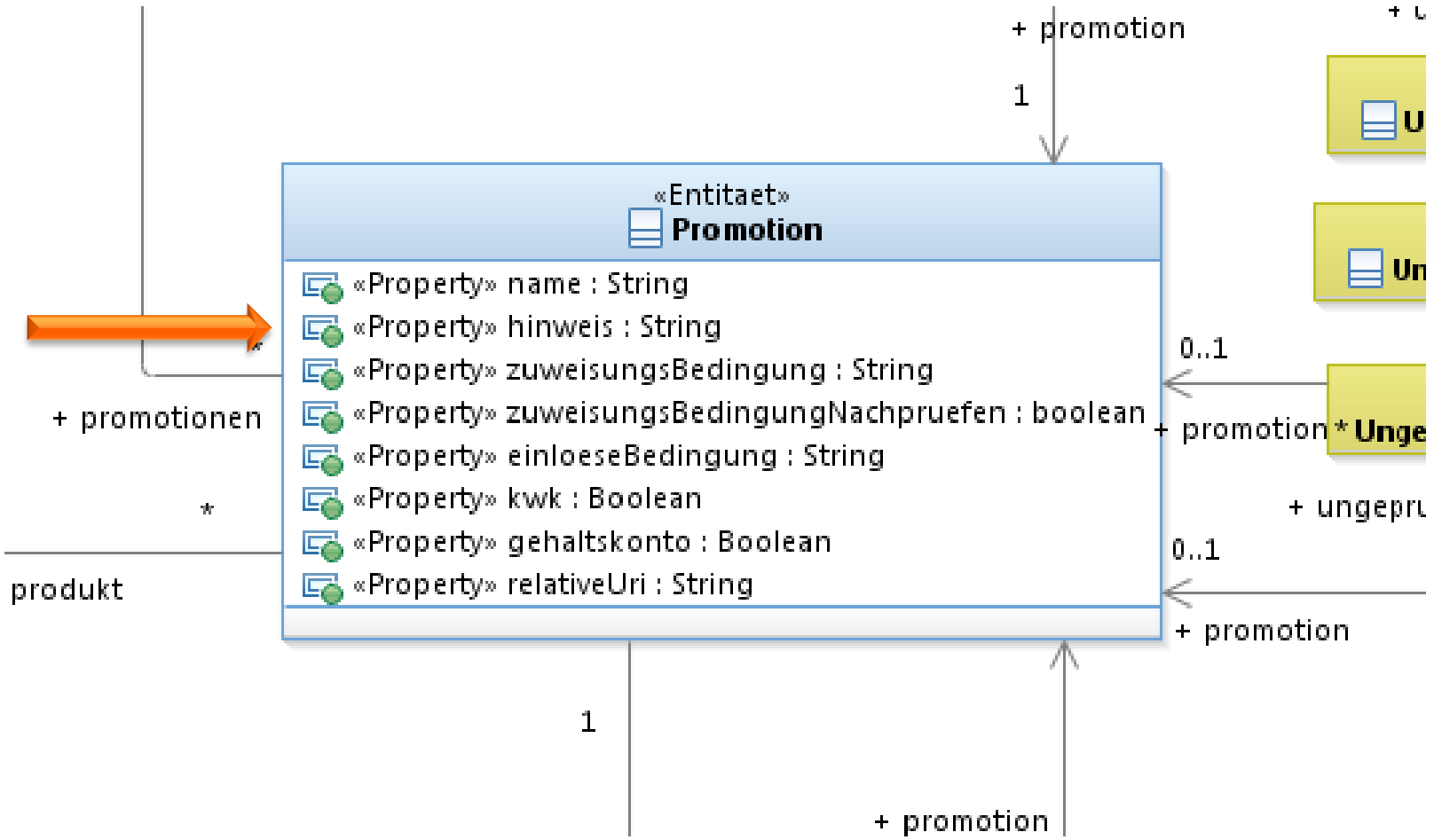
* Pflichtfelder

Weiter

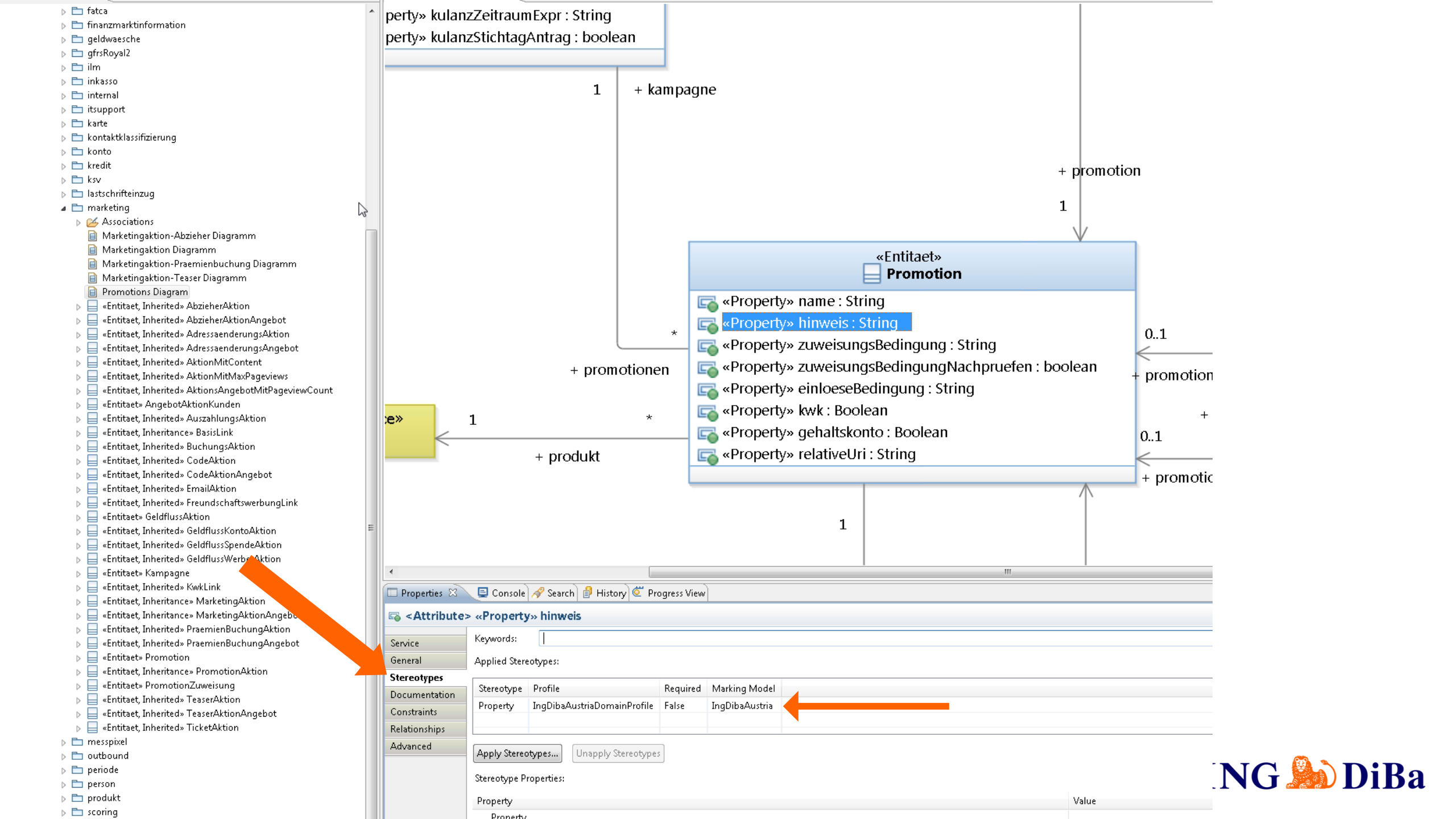
ING DiBa


TODO kulanzVon/ kulanzBis notwendig?
+ nachfolgendeKampagne
0..1
«Entitaet» Kampagne
«Property» name : String
«Property» gueltigVon : Date
«Property» gueltigBis : Date
«Property» kulanzVon : Date
«Property» kulanzBis : Date
«Property» kulanzZeitraumExpr : String
«Property» kulanzStichtagAntrag : boolean
1
+ kampagne
+ promotionen
«Entitaet, Inheritance» BasisLink
+ werber
NatuerlichePerson
«Entitaet, Inherited» KwkLink
«Property» code : String
«Property» /aktivWerbung : boolean
«Entitaet, Inherited» FreundschaftswerbungLink
+ kwkLink
+ ungepruefterSparAntrag
«Entitaet» UngepruefterSparAntrag
«Entitaet» UngepruefterKreditAntrag
+ ungepruefterKreditAntrag
+ ungepruefterGiroKontoAntrag
«Entitaet» UngepruefterGiroKontoAntrag
+ promotion
«Entitaet» Promotion
«Property» name : String
«Property» hinweis : String
«Property» zuweisungsBedingung : String
«Property» zuweisungsBedingungNachpruefen : boolean
«Property» einloeseBedingung : String
«Property» kwk : Boolean
«Property» gehaltskonto : Boolean
«Property» relativeUri : String
«Entitaet, Inheritance»
+ produkt
eine vorhandene Instanz des Antrags ist die Verbindung von PromotionZuweisung und BasisLink
«Entitaet, Inheritance» Antrag
+ antrag
+ ungepruefterAntrag
«Entitaet» Team
+ team
«Entitaet, Inherited» TicketAktion
«Property» wiedervorlage : Integer
«Property» bearbeitungsText : String
+ aktionen
+ promotionZuweisung
«Entitaet» PromotionZuweisung
«EnumProperty» status : StatusEnum
«Property» durchgefuehrtAm : Date
«Property» kulanz : Boolean
+ konto
«Entitaet» KundenKonto
«Entitaet, Inheritance» PromotionAktion
«Entitaet, Inherited» EmailAktion
«Property» subjectWebContent : String
«Property» htmlWebContent : String
«Property» textWebContent : String
«Entitaet, Inherited» AuszahlungsAktion
«Property» betrag : BigDecimal
«Property» text : String
«Entitaet, Inherited» BuchungsAktion
«Property» betrag : BigDecimal
«Property» text : String
«Property» buchungsSchluessel : long
«Entitaet» GeldflussAktion
«Property» betrag : BigDecimal
«Property» text : String
«Entitaet, Inherited» GeldflussSpendeAktion
«Property» empfaengerIban : String
«Property» empfaengerKontobez : String
«Property» empfaenger3ic : String
«Entitaet, Inherited» GeldflussWerberAktion
«Entitaet, Inherited» GeldflussKontoAktion
+ buchungsSchluessel
«Entitaet» BuchungsSchluessel

Neue Property hinzufügen



ING DiBa


fatca
finanzmarktinformation
geldwaesche
gfrsRoyal2
ilm
inkasso
internal
itsupport
karte
kontaktklassifizierung
konto
kredit
ksv
lastschrifteinzug
marketing
Associations
Marketingaktion-Abzieher Diagramm
Marketingaktion Diagramm
Marketingaktion-Praemienbuchung Diagramm
Marketingaktion-Teaser Diagramm
Promotions Diagram
«Entitaet, Inherited» AbzieherAktion
«Entitaet, Inherited» AbzieherAktionAngebot
«Entitaet, Inherited» AdressaenderungsAktion
«Entitaet, Inherited» AdressaenderungsAngebot
«Entitaet, Inherited» AktionMitContent
«Entitaet, Inherited» AktionMitMaxPageviews
«Entitaet, Inherited» AktionsAngebotMitPageviewCount
«Entitaet» AngebotAktionKunden
«Entitaet, Inherited» AuszahlungsAktion
«Entitaet, Inheritance» BasisLink
«Entitaet, Inherited» BuchungsAktion
«Entitaet, Inherited» CodeAktion
«Entitaet, Inherited» CodeAktionAngebot
«Entitaet, Inherited» EmailAktion
«Entitaet, Inherited» FreundschaftswerbungLink
«Entitaet» GeldflussAktion
«Entitaet, Inherited» GeldflussKontoAktion
«Entitaet, Inherited» GeldflussSpendeAktion
«Entitaet, Inherited» GeldflussWerb Aktion
«Entitaet» Kampagne
«Entitaet, Inherited» KwkLink
«Entitaet, Inheritance» MarketingAktion
«Entitaet, Inheritance» MarketingAktionAngebo
«Entitaet, Inherited» PraemienBuchungAktion
«Entitaet, Inherited» PraemienBuchungAngebot
«Entitaet» Promotion
«Entitaet, Inheritance» PromotionAktion
«Entitaet» PromotionZuweisung
«Entitaet, Inherited» TeaserAktion
«Entitaet, Inherited» TeaserAktionAngebot
«Entitaet, Inherited» TicketAktion
messpixel
outbound
periode
person
produkt
scoring
perty» kulanzZeitraumExpr : String
perty» kulanzStichtagAntrag : boolean
1
+ kampagne
+ promotion
1
«Entitaet»
Promotion
«Property» name : String
«Property» hinweis : String
«Property» zuweisungsBedingung : String
«Property» zuweisungsBedingungNachpruefen : boolean
«Property» einloeseBedingung : String
«Property» kwk : Boolean
«Property» gehaltskonto : Boolean
«Property» relativeUri : String
*
+ promotionen
0..1
+ promotion
:e»
1
*
+ produkt
0..1
+ promotic
1
Properties
Console
Search
History
Progress View
<Attribute> «Property» hinweis
Service
General
Stereotypes
Documentation
Constraints
Relationships
Advanced
Keywords:
Applied Stereotypes:
Stereotype
Profile
Required
Marking Model
Property
IngDibaAustriaDomainProfile
False
IngDibaAustria
Apply Stereotypes...
Unapply Stereotypes
Stereotype Properties:
Property
Value
Property

Stereotypes

Documentation

Constraints

Relationships

Advanced

Stereotype	Profile	Required	Marking Model
Property	IngDibaAustriaDomainProfile	False	IngDibaAustria

Apply Stereotypes... Unapply Stereotypes

Stereotype Properties:

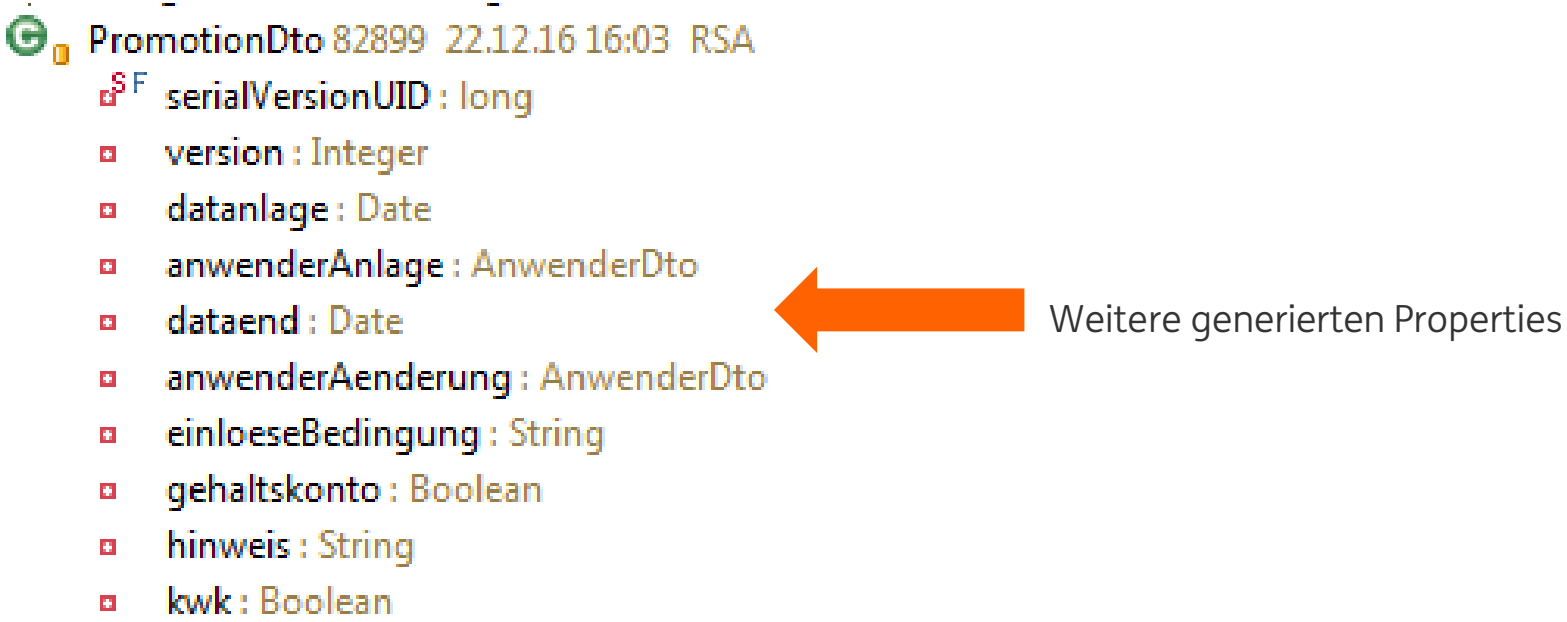
Property	Value
Property	
databasetype	2 - DEFAULT DATABASE TYPE
eager	False
foreignkeyspaltenname	
ignoreDtoSynchronization	False
ignoreNotFound	False
inverseforeignkeyspaltenname	
isPrimarykey	False
length	4000
migrationHint	
nullable	True
orderbyAttribut	
orderbyDirection	0 - ASC
referencedColumnName	
spaltenname	
temporalType	1 - DATE

Ergebnisse

Property in der generierten Klasse PromotionBase (inkl. Getter/Setter):

```
80   /**
81    * TODO: [MODEL] Im UML-Modell kommentieren.
82    */
83   @Column(name = "PROM_HINWEIS", length = 4000)
84   private String hinweis;
```

Im Dto:



ING DiBa

Generierte Klassen/Interfaces

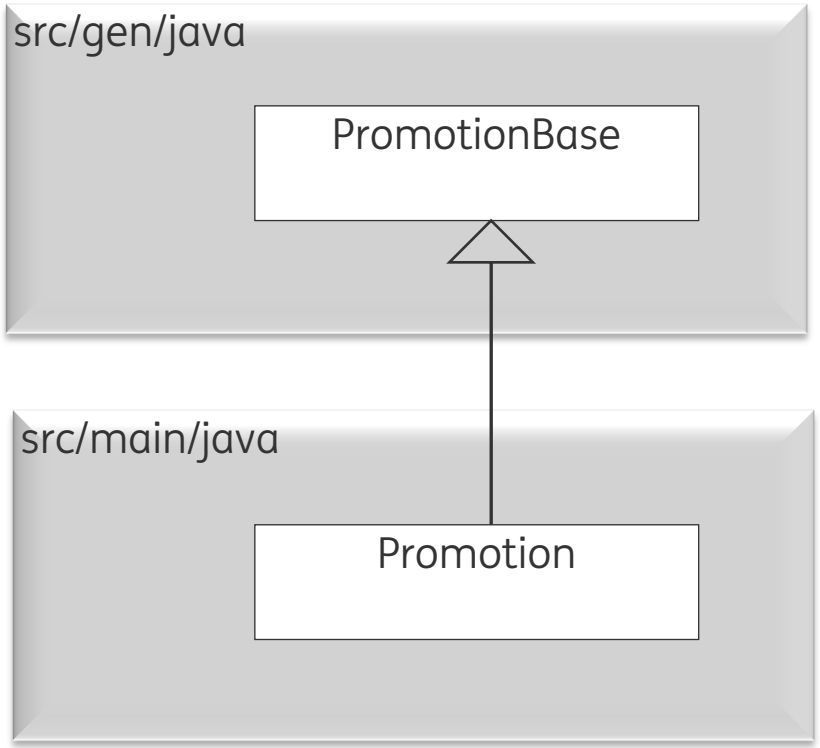
- Promotion
- PromotionBase
- PromotionDto

Durch händische Implementierung erweiterbar

- PromotionManager
- PromotionManagerImpl
- PromotionManagerBase
- PromotionManagerBaseImpl

Durch händische Implementierung erweiterbar

Trennung von generierten Klassen



Wird bei der Ausführung des Code-Generators überschrieben

Nicht generierter Code bleibt erhalten

Beispiel Promotion Klasse:

```java
package at.ing.diba.domain.marketing;

import javax.persistence.Entity;
import javax.persistence.Table;

@Entity
@Table(name = "PROMOTION")
public class Promotion extends PromotionBase {

}
```

- Promotion Klasse wird einmalig generiert
- In den Bereich src/main/java verschoben zwecks manueller Implementierungen

Schema-Upgrade: Anpassung der Datenbank

Generiert durch hibernate hbm2ddl

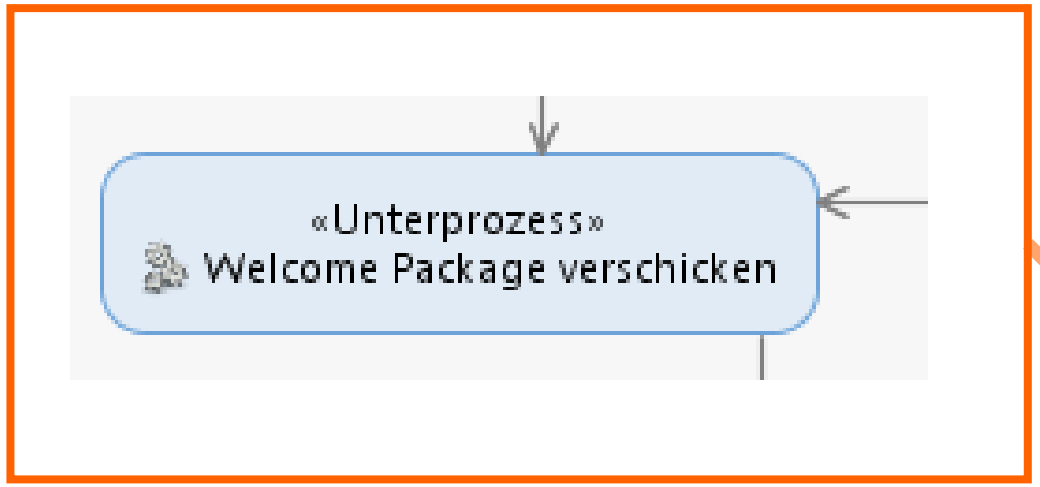
```
alter table Promotion add  PROM_HINWEIS varchar2(4000);
```

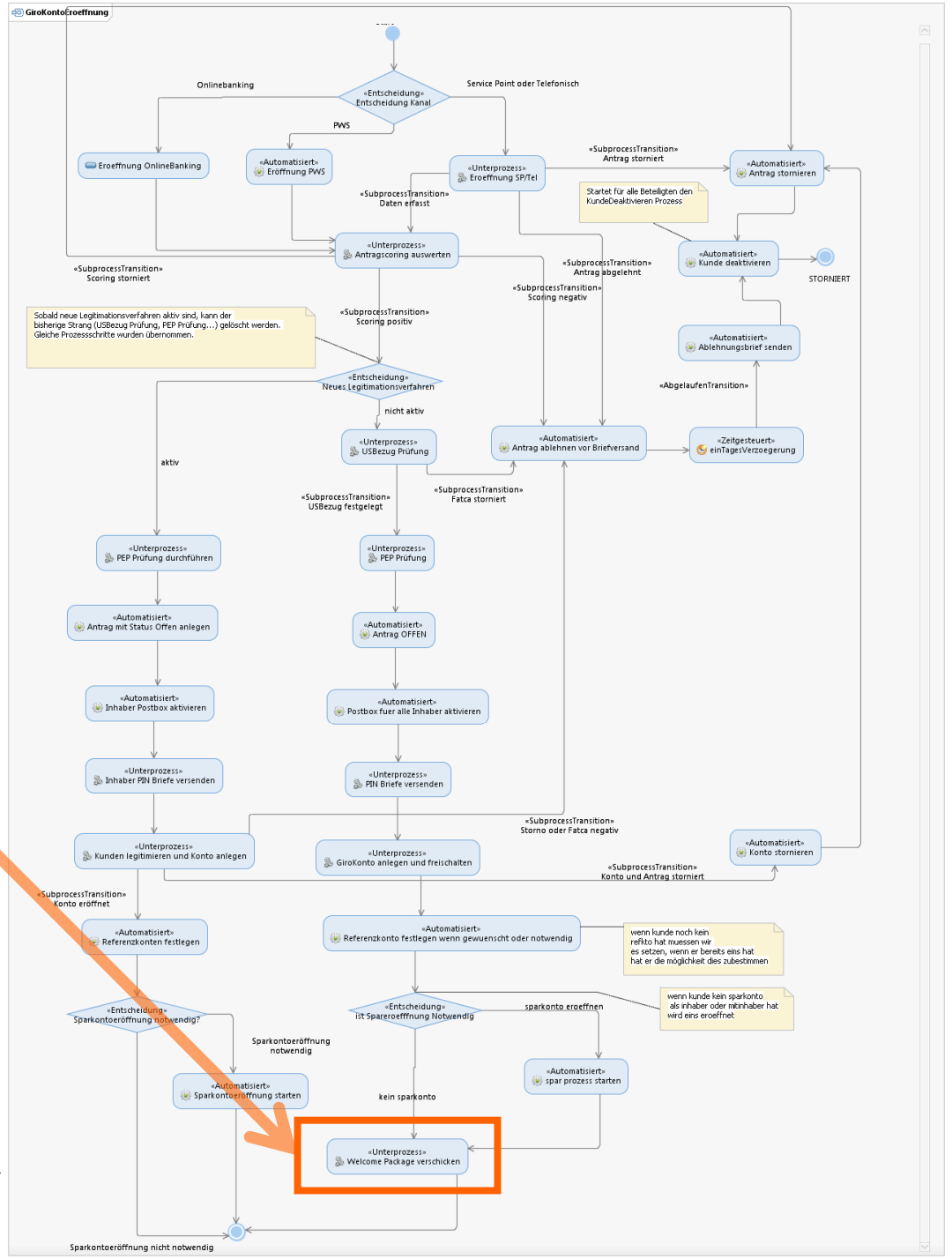
Beispiel: Modellierung von Prozessen

Anforderung:

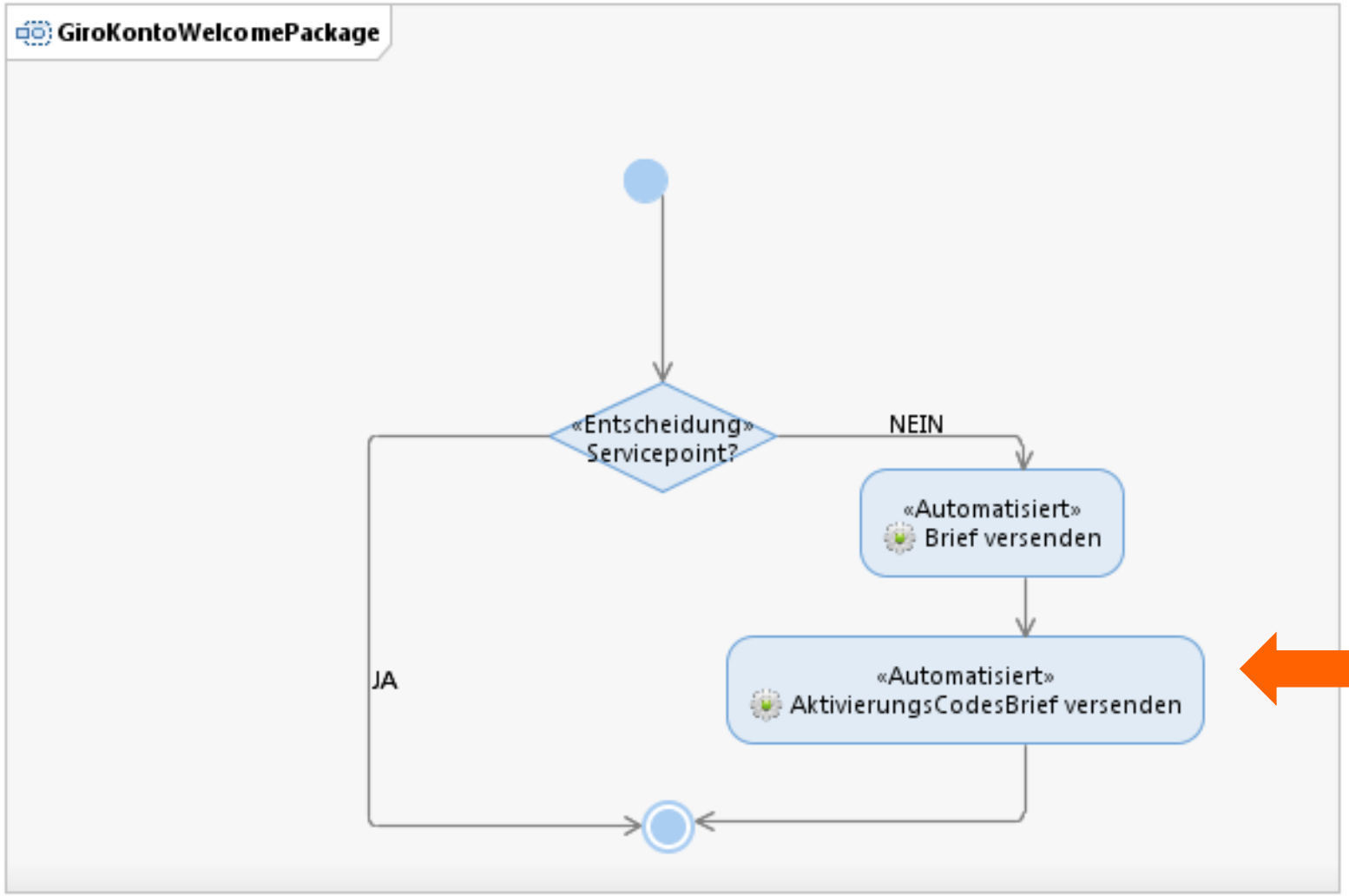
Im Eröffnungsprozess im Prozessschritt "WelcomePack", sollen die AktivierungsCodes generiert werden

Eröffnungsprozess





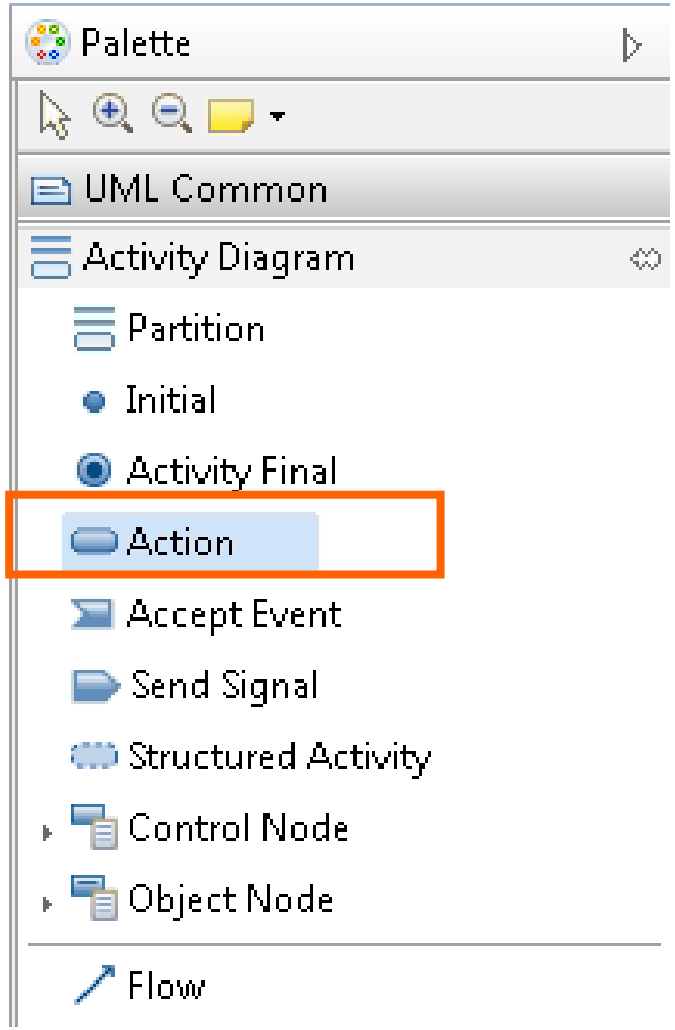
Prozess: GirokontoWelcomePackage

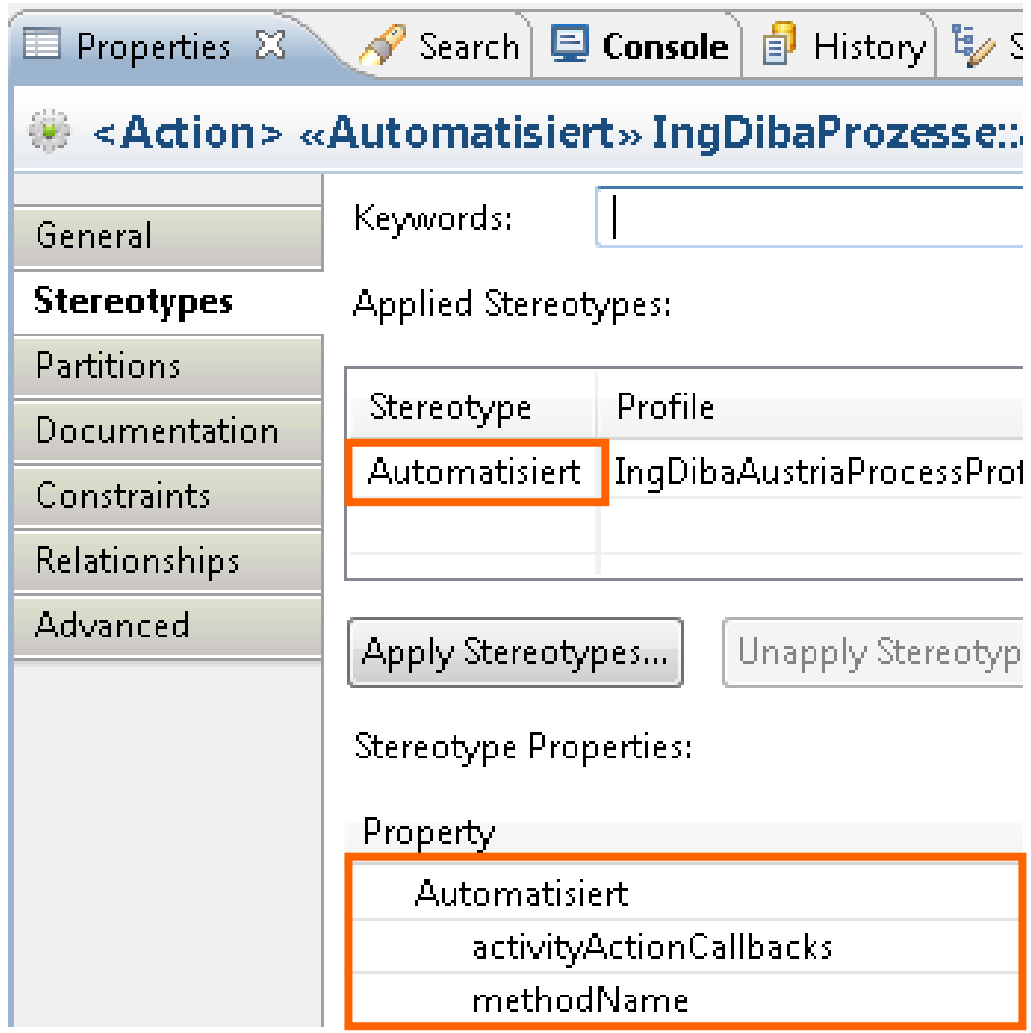


Neue Prozess Aktivität

ING DiBa

Neue Aktion anlegen





ING DiBa

Generierte Klassen für den GirokontoWelcomePackage-Prozess

- GirokontoWelcomePackage**ProcessDeployer**
  - Definition des Prozessflusses
  - Verknüpft Aktivitäten und Transitionen

- GirokontoWelcomePackage**ProcessConstant**
  - Definitionen der einzelnen Aktivitäten, Transitionen

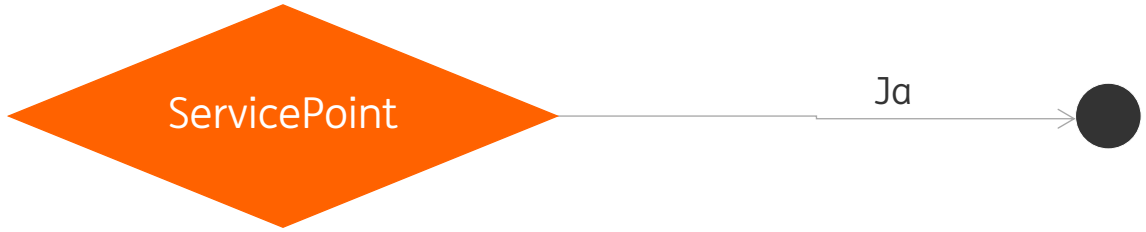
- GirokontoWelcomePackage**ProcessHandler**
  - Interface zwischen programmierten Funktionen und der Process Engine

- GirokontoWelcomePackage**ProcessHandlerImpl**
  - Notwendige Implementierung des Entwicklers

GirokontoWelcomePackageProcessDeployer

```
42  /* --------  ADD TRANSITION -------------- */
43
44    @Override
45  public void addTransitions(ProcessDefinitionBuilder processDefinitionBuilder) {
46
47    processDefinitionBuilder.addTransition(TransitionsEnum.JA,ActivitiesEnum.SERVICEPOINT,ActivitiesEnum.END, null);
48
```

Beispiel für eine Verknüpfung von Transitionen und Activities



ING DiBa

GirokontoWelcomePackageProcessConstant

```
20  /* --------  ACTIVITYS ENUM ------------- */
21
22  public enum ActivitiesEnum implements Activities
23  {
24    START,
25    SERVICEPOINT,
26    END,
27    BRIEF_VERSENDEN,
28    AKTIVIERUNGSCODESBRIEF_VERSENDEN;
29    @Override
30    public String getName() {
31      return name();
32    }
33
34    @Override
35    public String getProcessDefinitionName() {
36      return PROZESS_NAME;
37    }
38  }
```

```
41  /* --------  TRANSITIONS ENUM ------------- */
42
43  public enum TransitionsEnum implements Transitions
44  {
45  JA
46  ,TRANSITION1
47  ,NEIN
48  ,TRANSITION2
49  ,TRANSITION3
50  ;
51  }
52
53 }
```

GirokontoWelcomePackageProcessHandler

```
1
2 package at.ing.diba.process.giro.kontoeroeffnung;
3
4 import at.ing.diba.processengine.api.ProcessHandler;
5 import at.ing.diba.business.BusinessException;
6
7 public interface GiroKontoWelcomePackageProcessHandler extends ProcessHandler
8 {
9     GiroKontoWelcomePackageProcessConstant.TransitionsEnum servicepoint() throws BusinessException;
10
11     void briefVersenden() throws BusinessException;
12
13     void aktivierungscodesbriefVersenden() throws BusinessException;
14
15 }
```

GirokontoWelcomePackageProcessHandlerImpl

```
46  @Override
47  public void aktivierungscodesbriefVersenden() throws BusinessException {
48
49    GiroKontoEroeffnungContext ctx = getContext();
50    NatuerlichePerson inhaber = ctx.getAntragSteller();
51    long idPerson = inhaber.getId();
52    List<AktivierungsCode> code = codeManager.findByPersonAbLaufNummer(idPerson, 0);
53
54    if (code.size() == 0) {
55      personAutorisationService.generiereUndVersendeNeueCodes(idPerson, false);
56    }
57
58  }
```

Referenzen

http://www.inf.fu-berlin.de/inst/ag-se/teaching/S-BSE/135_kuehn_MDA.pdf
https://www.gi.de/service/informatiklexikon/detailansicht/article/model-driven-architecture.html
https://de.wikipedia.org/wiki/Generative_Programmierung
https://de.wikipedia.org/wiki/Modellgetriebene_Architektur
http://xml.coverpages.org/OMG-MDAFAQfinal1.pdf
http://www.omg.org/mda
http://www.software-kompetenz.de/?5677