Annotation based presentation models for view rendering and data binding with Linkki and Vaadin

jan.ortmann@faktorzehn.de

Annotation based presentation models for view rendering and data binding with Linkki and Vaadin

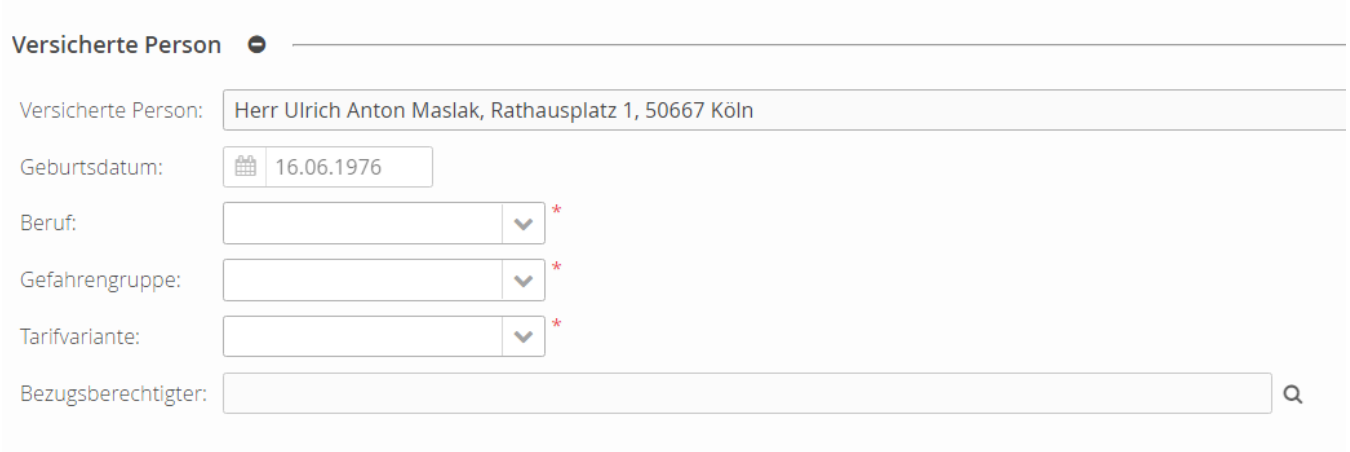
Annotation based presentation models for view rendering and data binding with Linkki and Vaadin

Annotation based presentation models for view rendering and data binding with Linkki and Vaadin

or

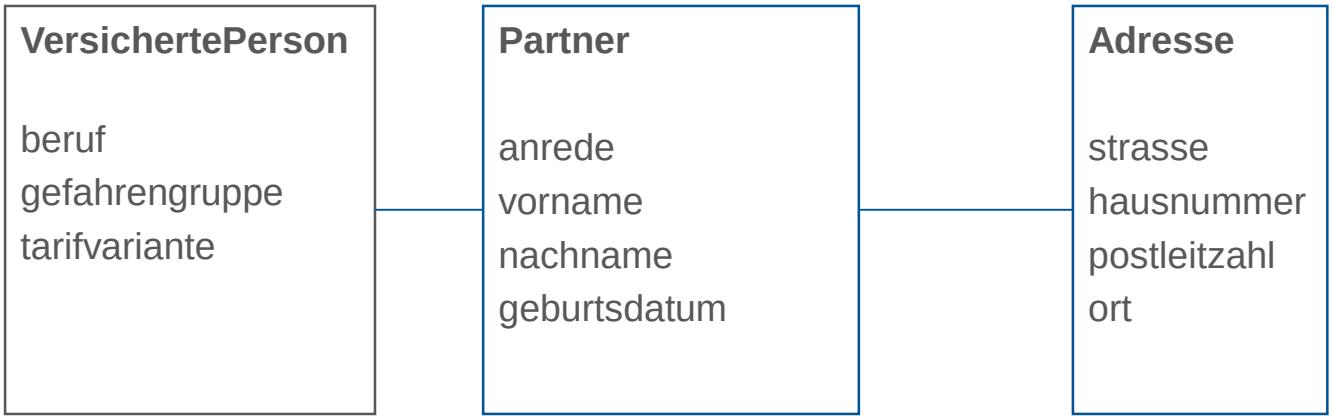
How to code UIs really, really fast!

In Anwendungen müssen Werte aus dem Domain Model in die UI übertragen werden und umgekehrt.

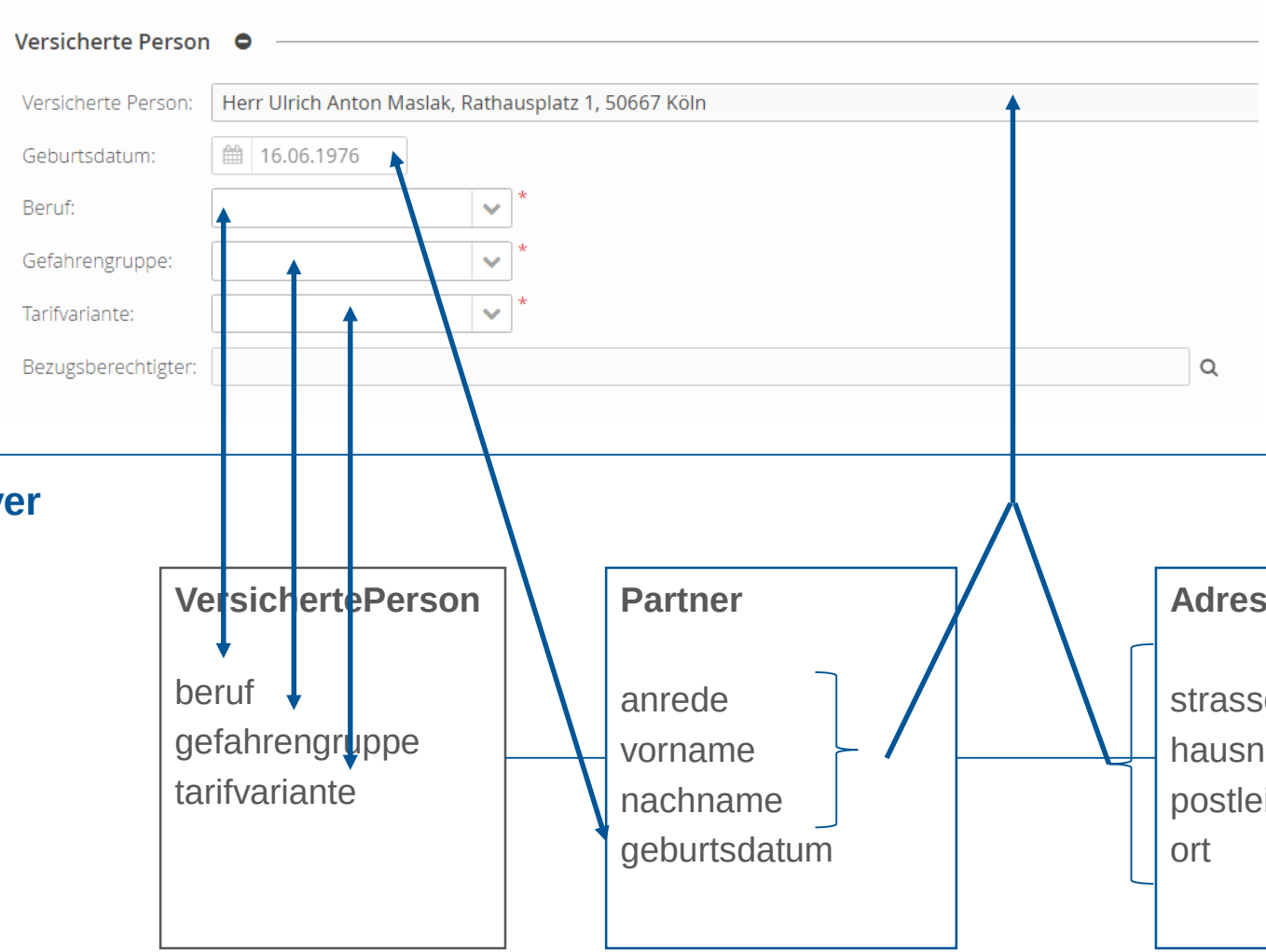


UI

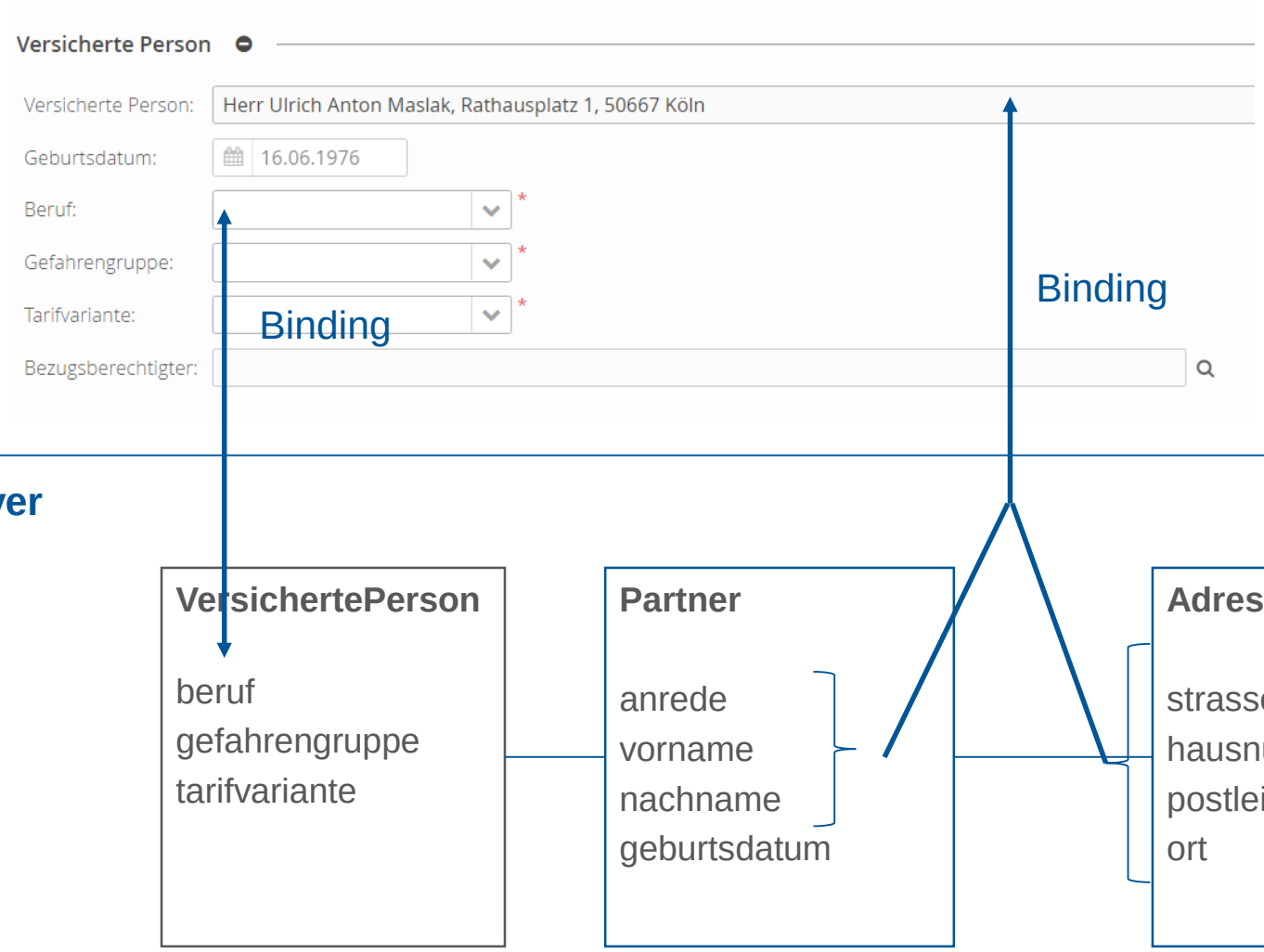
Domain Layer



In Anwendungen müssen Werte aus dem Domain Model in die UI übertragen werden und umgekehrt.

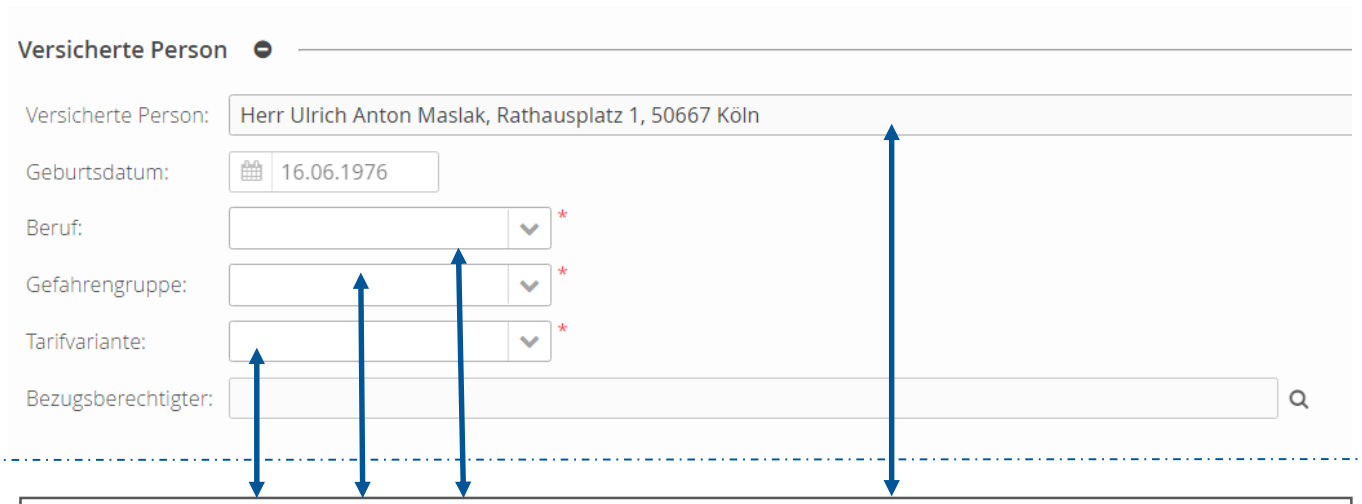


Databinding: Verknüpfen von Eigenschaften von Modelobjekten mit UI-Controls mit einem Mechanismus zur Synchronisierung.



Presentation Model Objects / View Model

Represent the state and behavior of the presentation independently of the GUI controls used in the interface. (http://martinfowler.com/eaaDev/PresentationModel.html)



Controls

View Model

VersichertePersonSectionPmo

UI

Domain Layer

Service
Operation
Repository
…
Any other domain layer class

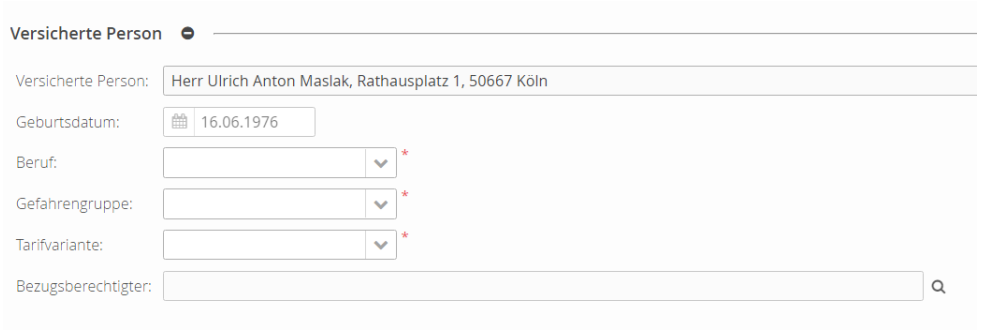
VersichertePerson
<<Entity>>
beruf
gefahrengruppe
tarifvariante

Partner
<<Entity>>
anrede
vorname
nachname
geburtsdatum

Adresse
<<Entity>>
strasse
hausnummer
postleitzahl
ort

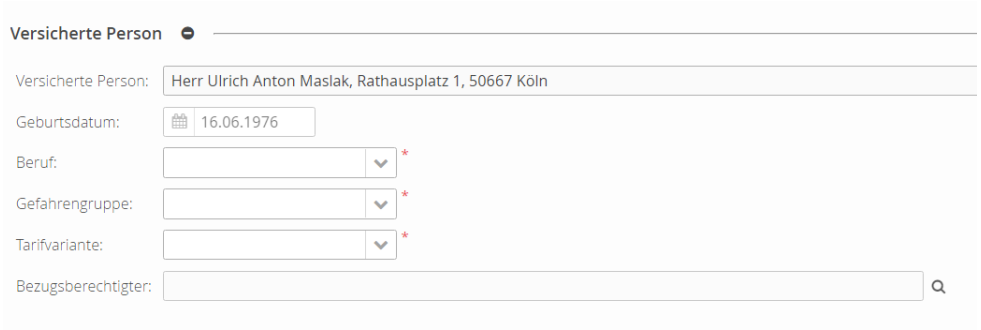
So könnte das PMO aussehen …

```
public class VersichertePersonSectionPmo {

    private final UvVersichertePerson vp;

    public Beruf getBeruf() {
        return vp.getBeruf();
    }
    public void setBeruf(Beruf beruf) {
        vp.setBeruf(beruf);
    }
    …
}
```



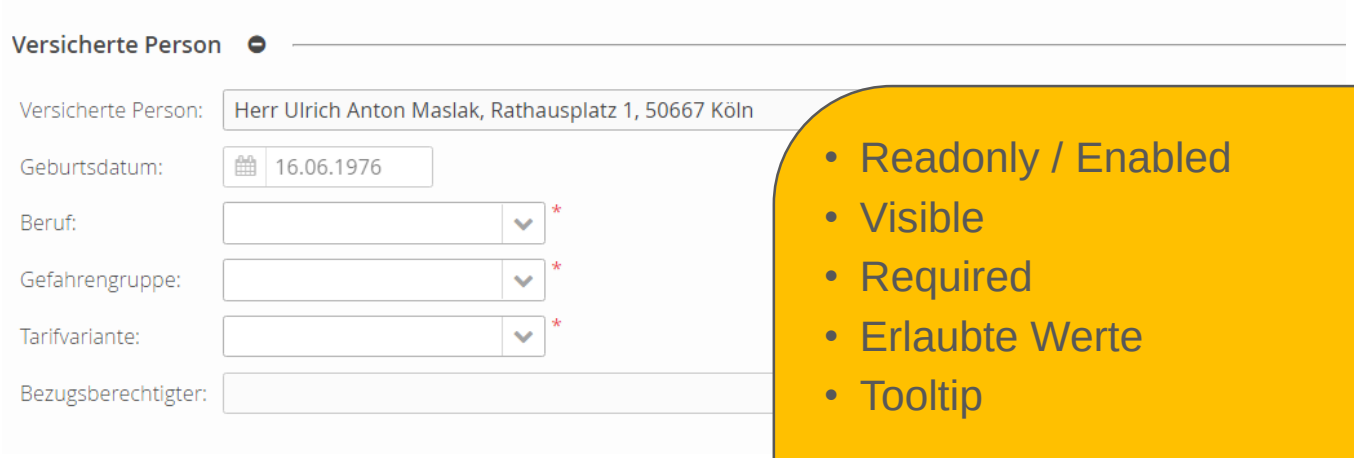
So könnte das PMO aussehen …

```
public class VersichertePersonSectionPmo {

    private final UvVersichertePerson vp;

    public Beruf getBeruf() {
        return vp.getBeruf();
    }
    public void setBeruf(Beruf beruf) {
        vp.setBeruf(beruf);
    }

    public String getVersichertePerson() {
        Partner partner = vp.getPartner();
        Address address = partner.getDefaultAddress();
        return partner.getAnrede() + ", " + partner.getVorname() + ", „ …
    }

    public LocalDate getGeburtsdatum() {
        return vp.getPartner().getDateOfBirth();
    }
}
```



Kurzer Blick in ein reales System …

Auch andere Aspekte von UI Komponenten können / sollten per Data Binding berücksichtigt werden:



- Readonly / Enabled
- Visible
- Required
- Erlaubte Werte
- Tooltip

Die Fachlogik steht idR entweder in den Domain Model Klassen oder in anderen Klassen des Domainy Layers zur Verfügung.

UI

Domain Layer

Service
Operation
Repository
…
Any other domain layer class

VersichertePerson

beruf
gefahrengruppe
tarifvariante

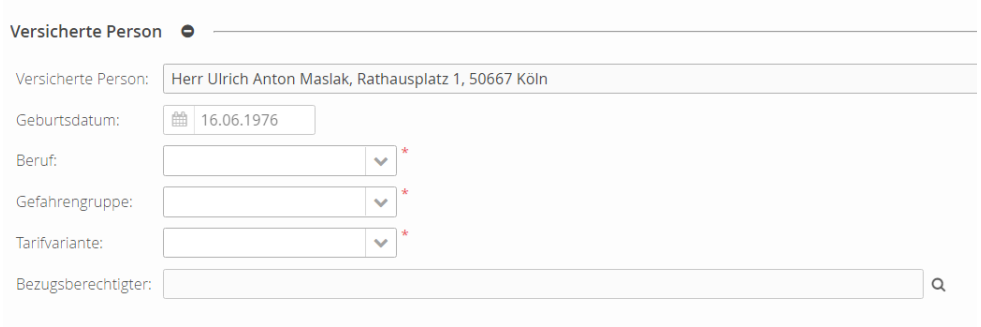
Partne

anrede
vorname
nachname
geburtsdatum

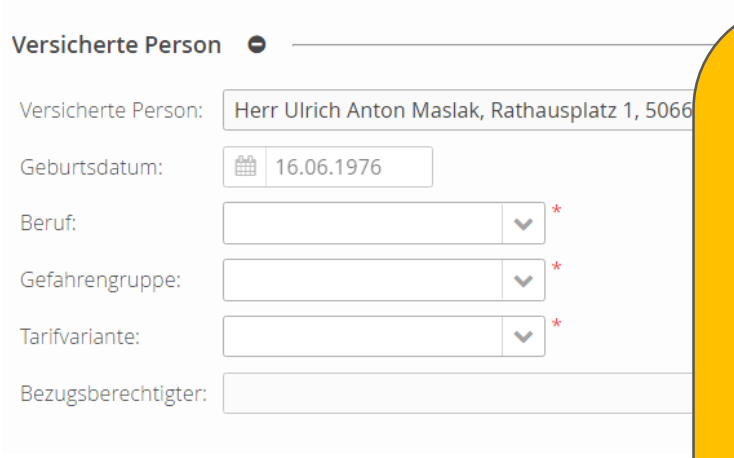
postleitzahl
ort

So könnte das PMO für das enablen / disablen des Bezugsberechtigten erweitert werden …

```
public class VersichertePersonSectionPmo {

    private final UvVersichertePerson vp;
      ….

    public boolean isBezugsberechtigterEnabled() {
        return vp.getTodesfallLvb() != null;
    }

}
```



Herausforderung: Berücksichtigung der Abhängigkeit zwischen Feldern.

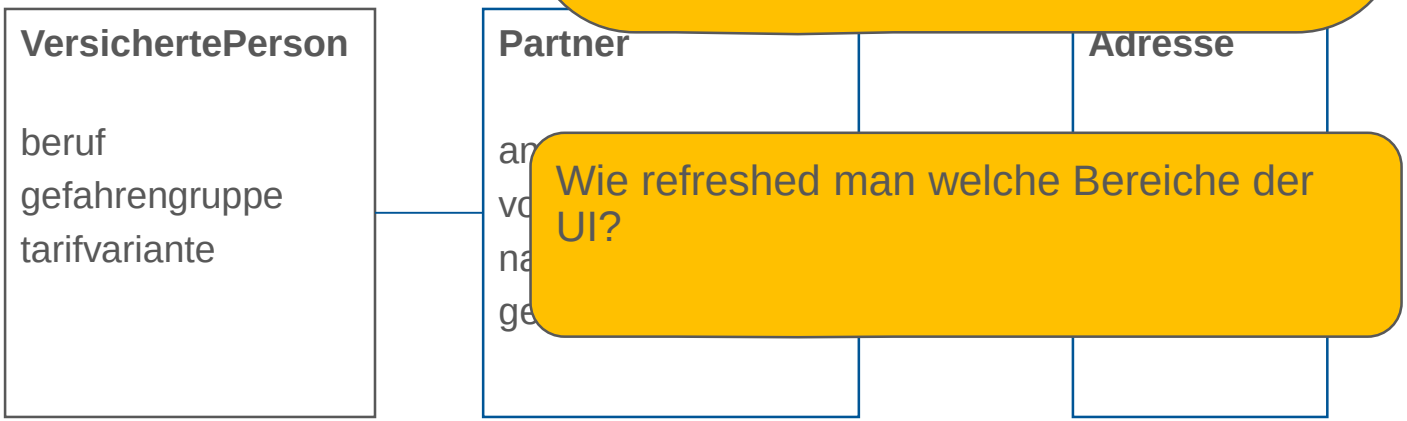


UI

Szenario:

1. Der Benutzer ändert den Wert in einem Feld.
2. Dadurch ergeben sind neue Werte in anderen Feldern.
3. Die erlaubten Werte in Comboboxen ändern sind.
4. Der enabled / disabled State anderer Felder ändert sich.
5. …

Domain Layer



Linkki - Idee

Unterteile das User Interface von Geschäftsanwendungen in Pages und jede Pages in Sections.

Verwende einige Standard-Layouts für die Sections.

Schreibe für jede Section ein PMO und annotiere es mit Anweisungen, wie die Section gerendert werden soll.

Linkki rendert die Sections auf Basis der annotierten PMOs und führt das Databinding durch. Nach jeder Änderung werden die Bindings auf einer Page refreshed.

Und nun zur Praxis …

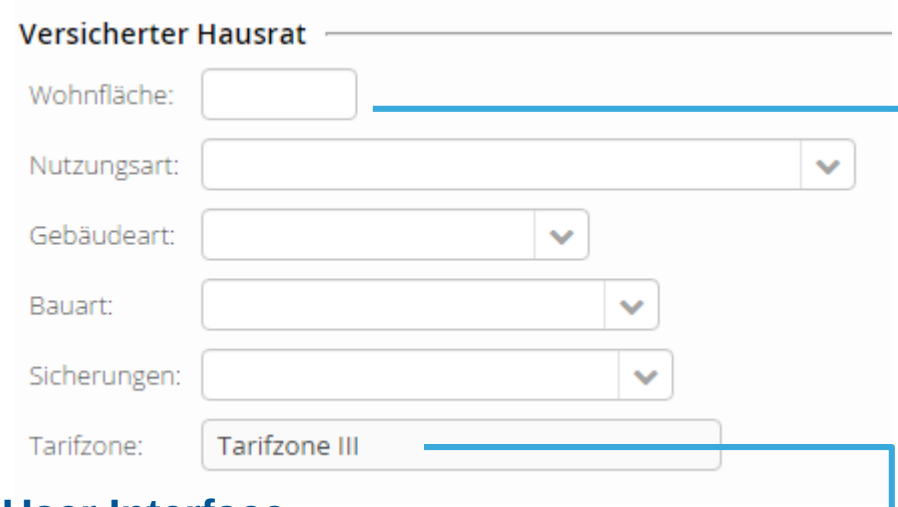
Anzeige von Fehlermeldungen

Anforderungen:

Fachliche Prüfungen sollen in der Geschäftslogik implementiert werden, ohne Bezug zum UI.

Fehlerhafte Felder sollten markiert werden.

Durch das Data Binding gibt es eine Zuordnung der Modellobjekte und ihrer Eigenschaften zu Eingabefeldern



User Interface

Domain Layer

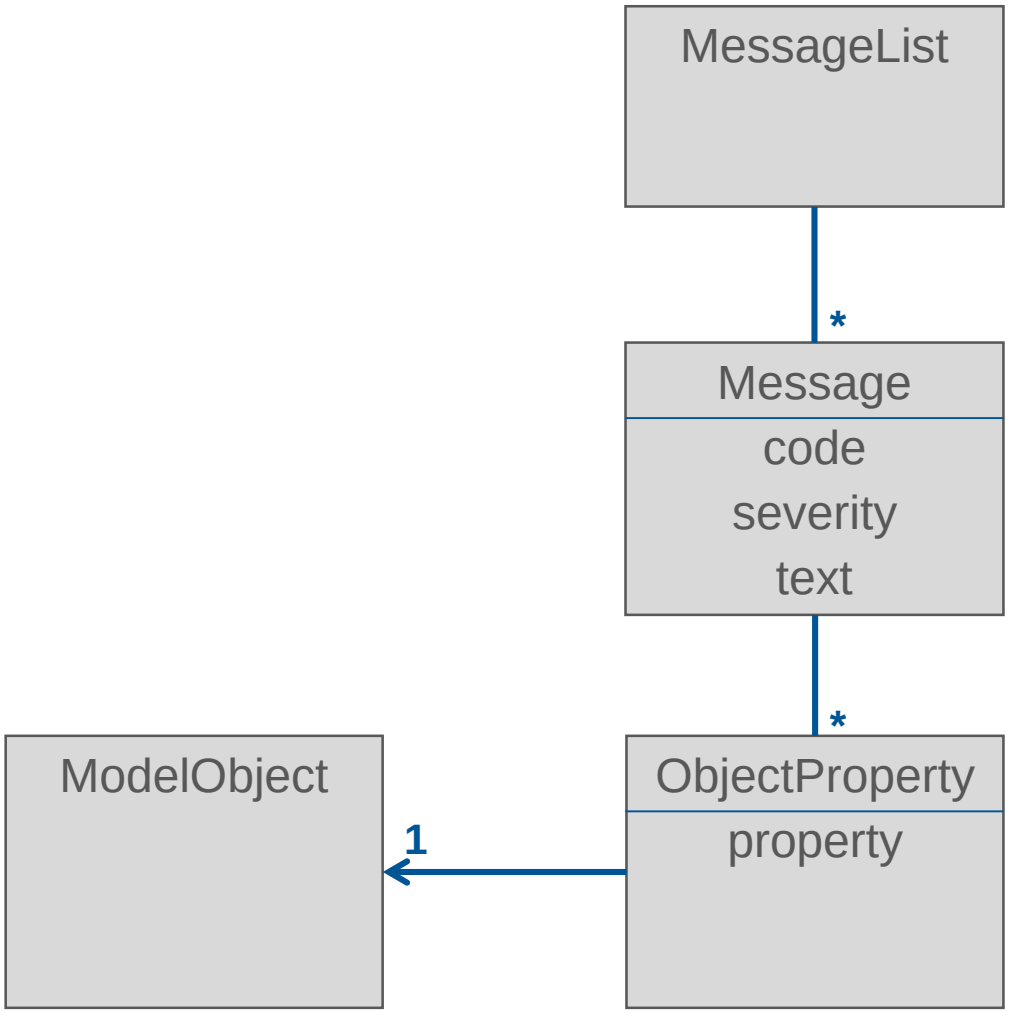
Versichertes-Gebäude

tarifzone

Gebäude

Wohnfläche
Nutzungsart
Gebäudeart
Bauart
Sicherungen

ValidationService liefert eine Liste von Messages zurück

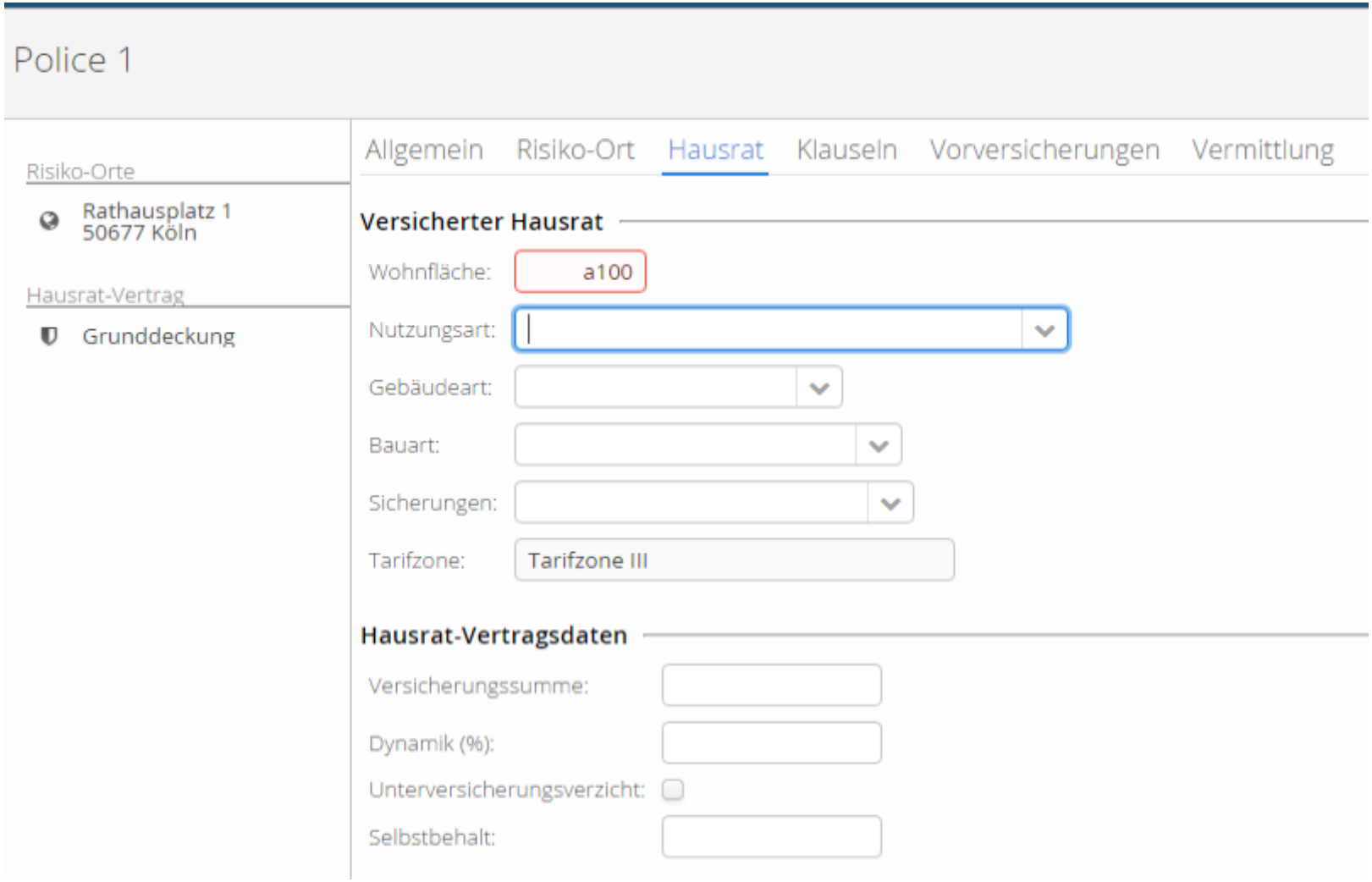


Jede Message weiß auf welche Eigenschaften welcher Objekte sich die Nachricht bezieht.

Einfachster Fall:

Eine Eigenschaft eines Objekts ist fehlerhaft.

Damit können die Felder als fehlerhaft markiert werden und auch einer Seite zugeordnet werden



Und nun zur Praxis …

What else?

Tabellen können genauso einfach mit PMOs erstellt werden.

- 1 PMO für die Tabelle, 1 PMO für die Zeilen.

Einfache Dialog können mit PMOs erstellt werden.

Definition von Querschnittsaspekte möglich

Beispielsweise alles readonly, wenn keine Berechtigung vorhanden.

Wenn die Standard-Layouts nicht passen, können einzelne Sections individuell designed werden und die Controls an ein PMO gebunden werden (ebenfalls per Annotation).

Fazit

User Interface für Geschäftsanwendungen im Back-Office können extrem schnell entwickelt werden.

Klare Strukturierung des UI Codes.

- 1 PMO pro Section
- 2 PMOs pro Tabelle
- Klare Trennung von Layout, Inhalt der UI und Geschäftslogik.

Es reichen grundlegende Java Kenntnisse, um ein UI zu bauen.

Vaadin erlaubt es einem, so ein Framework zu bauen.