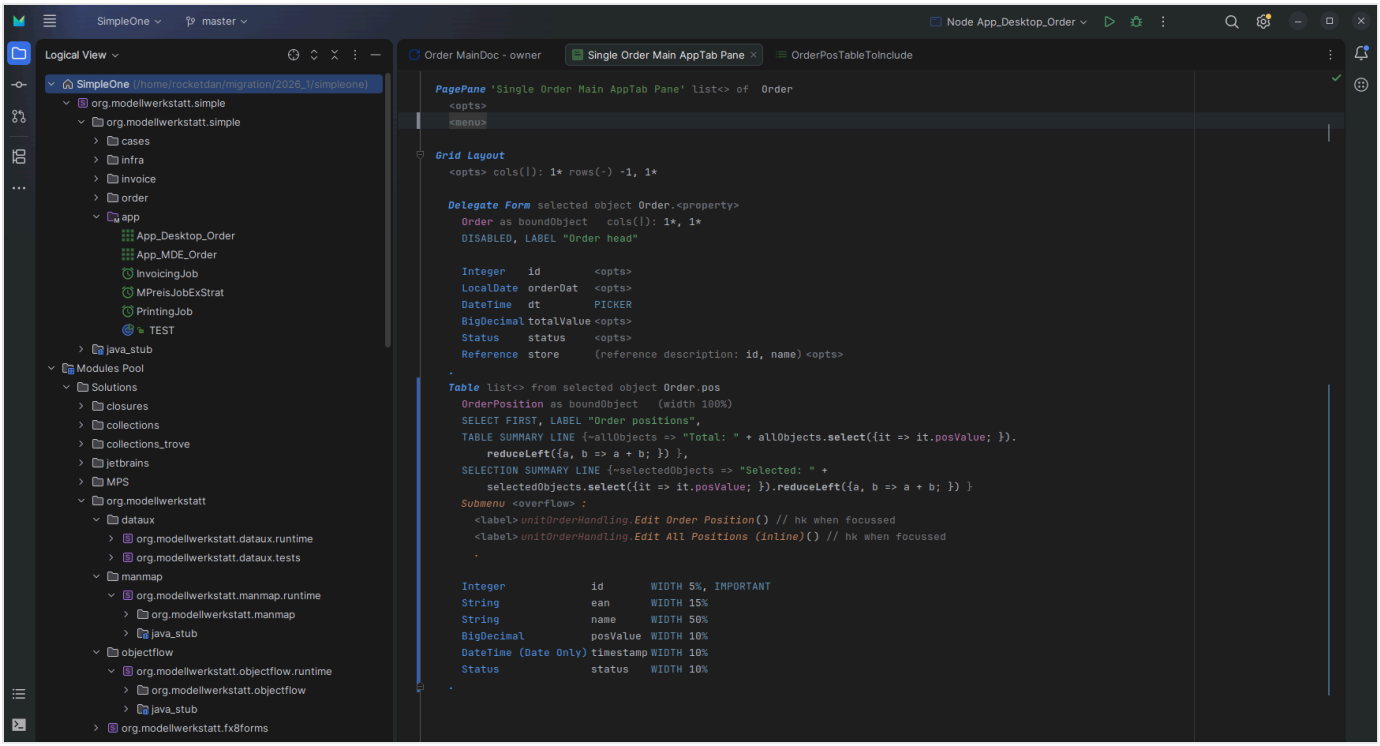
modellwerkstatt

MoWare-Werkbank

Geschäftsanwendungen **modellieren**, nicht von Grund auf programmieren.

Die MoWare-Werkbank ist eine Java-basierte Werkzeugkette für die schnelle Entwicklung individueller Geschäftsanwendungen. Drei eng integrierte domänenspezifische Sprachen beschreiben Persistenz, Geschäftslogik und Benutzeroberflächen. Generatoren und Laufzeitumgebungen übernehmen den Infrastruktur-Code, und Sie konzentrieren sich auf das Fachliche.



DataUX im MPS-Editor: Eine PagePane mit Formular und Tabelle in einer lesbaren Projektion, kein Boilerplate-Code.

GESCHÄFTSLOGIK

ObjectFlow

`org.modellwerkstatt.objectflow`

Fachliche Datenstrukturen, Services und Commands für Anwendungsabläufe, dazu Testsuiten, Rollen und Berechtigungen.

`Entity · Value`
`Object · DTO · Service · Command`

PERSISTENZ

ManMap

`org.modellwerkstatt.manmap`

Mapping zwischen Tabellen und Entities, Laden, Speichern und Löschen über Repositories sowie SQL-Abfragen direkt in DTOs.

`Mapping · Repository · SQL · Read Model`

OBERFLÄCHE

DataUX

`org.modellwerkstatt.dataux`

Tabellen und Formulare über Spalten, Felder, Formate und Datenbindung, dazu Layouts, Anwendungen mit Hauptmenü und Batchjobs.

`Table · Form · Layout · AppUI · BatchJob`

WAS DIE WERKBANK LEISTET

01

Fachlichkeit im Mittelpunkt
Datenstrukturen und Geschäftsregeln bleiben die zentrale Entwurfsaufgabe und im Modell sichtbar.

02

Ausdrucksstarke Logik
Berechnungen und Prüfungen direkt an den Daten, mit Java-Ausdrücken und Mengenoperationen.

03

Einheitliche Architektur
Klare Zuständigkeiten für Daten, Logik und UI erleichtern Orientierung und Zusammenarbeit.

04

Investitionssicherheit
Modelle überdauern Frameworks: Technik wird zentral in Generatoren und Laufzeiten angepasst.

ExpensiveCode & CheapCode

MoWare unterscheidet bewusst zwischen aufwendig erarbeitetem Fachwissen und leicht anpassbaren Teilen einer Anwendung. So bekommt jeder Teil genau die Sorgfalt, die er braucht.

FACHLICHER KERN

ExpensiveCode

Datenstrukturen, Geschäftsregeln und Verarbeitungslogik. Ihre Entwicklung erfordert Domänenwissen, und Fehler zeigen sich oft erst in der fachlichen Prüfung. Deshalb wird sorgfältig modelliert und mit Testsuiten abgesichert.

`Sorgfalt & Abstimmung`

OBERFLÄCHE & BEDIENUNG

CheapCode

UI-Beschreibungen, Menüs und Interaktion mit dem Modell. Fehler fallen beim Ausprobieren sofort auf, und Oberflächen werden in kurzen Feedbackzyklen schrittweise verbessert, ohne den Kern zu berühren.

`Schnelle Iteration`

Vom Modell zur laufenden Anwendung

ObjectFlow
Daten, Regeln, Anwendungsfälle

ManMap
Relationale Persistenz

DataUX
Formulare, Tabellen, Module

→

GENERATOR
Java-Code
aus dem Modell erzeugt

OFXCONFIG
wählt per AppFactory die Laufzeit

→

DESKTOP
JavaFX
Rich Client für Desktop-PCs

WEB
Vaadin auf Tomcat
Webanwendung im Browser

MOBIL
HTML5 auf Tomcat
MDE-Geräte (z. B. Zebra, Datalogic) und Smartphones

Ein Modell, mehrere Ziele: Anwendungsmodelle sind mit wenigen Ausnahmen zwischen allen drei Laufzeiten portabel. **Batchjobs** laufen direkt, zeitgesteuert per Cron oder kontinuierlich als Servlet, auf Wunsch auch mit Oberfläche.

Grundprinzipien

01
Das fachliche Modell ist der Ausgangspunkt
Regeln gehören an die Daten oder in Services, nicht in Oberfläche oder Persistenzcode.

02
Objektgraphen werden explizit geladen
Kein Lazy Loading und keine Magie: Was geladen wird, ist Teil des Entwurfs.

03
Auch Persistenz ist explizit
Check-in und Delete legen genau fest, was gespeichert oder gelöscht wird.

04
Lesen und Bearbeiten sind getrennt
Suchen und Auswertungen arbeiten mit read-only DTOs, Bearbeitung im eigenen Kontext.

05
Transaktionen folgen dem Anwendungsfall
Speichern erst beim erfolgreichen Abschluss, ein Abbruch hinterlässt keine Spuren.

06
Fachwissen bleibt technikunabhängig
Modelle sind frei von UI-Frameworks und überprüfbar durch Beispiele und Tests.

KONTAKT

Lassen Sie uns reden.

Wir bieten Beratung, Schulung und Support rund um die Werkbank, modellgetriebene Entwicklung und JetBrains MPS.

WEB	**www.modellwerkstatt.org**
MAIL	**info@modellwerkstatt.org**
ORT	**Innsbruck, Österreich**
QUELLEN	`github.com/danielstieger`

Stand Oktober 2026