vaadin + CDI = MVP

Oliver.Damm@akquinet.de

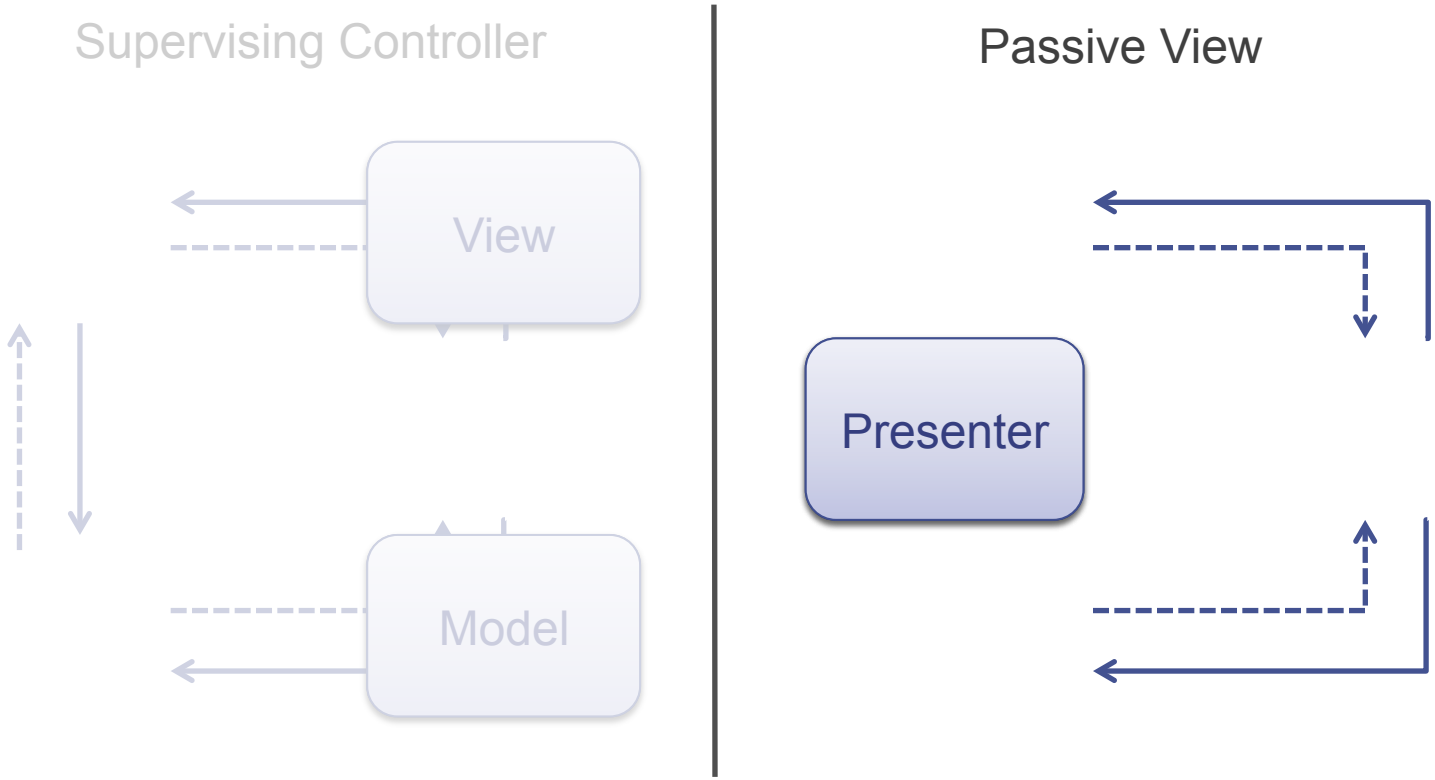
September 2013

Agenda

- Model View Presenter
- Context and Dependency Injection
- Demo Anwendung
- MVP im Code
- Entkopplung durch Events
- Testen der Logik

Copyright © 2013 – akquinet AG

Model View Presenter



Direkte Verbindung

Indirekte Verbindung (z.B. über Observer, Data Binding)

Copyright © 2013 – akquinet AG

Context and Dependency Injection

- Injizieren von Abhängigkeiten über Standard Java Annotationen
  - `@Inject`
- Abhängigkeiten werden Kontext-abhängig injiziert
  - `@SessionScope, @RequestScope, @Dependent`
  - Eigene Kontext-Implementierung möglich
- Volle Integration in Java EE 6
  - Glassfish, JBoss etc.
  - Tomcat, Jetty etc.

Copyright © 2013 – akquinet AG

Demo Anwendung

- Kontaktverwaltung
- Maven Projekt
- Sourcen auf GitHub

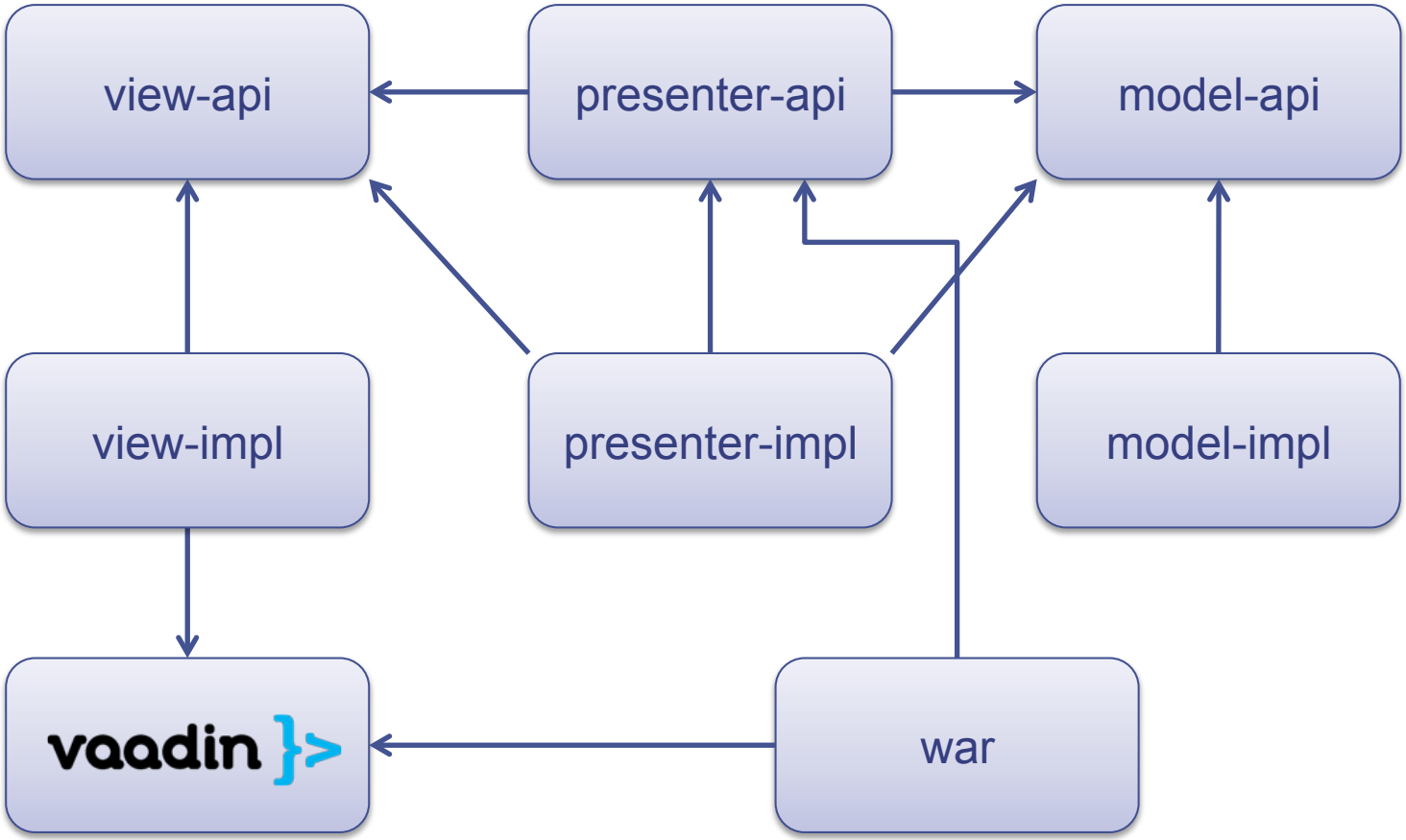
`https://github.com/akquinet/vaadin-cdi-mvp`

Copyright © 2013 – akquinet AG

Demo Anwendung

Copyright © 2013 – akquinet AG

Maven Projektstruktur



Copyright © 2013 – akquinet AG

API

- Interface `View`
- Interface `ObservableView`
- Konkretes Interface pro konkreten View

Implementierung

- Interface `VaadinView`
- `@Dependent`

Copyright © 2013 – akquinet AG

API

- Interface `Presenter`
- 1:1 Beziehung zu einem View
- Konkretes Interface pro konkreten Presenter

Implementierung

- Betreibt den View (API, UI Events)
- `@SessionScoped`

Copyright © 2013 – akquinet AG

API

- Schnittstelle zur Business-Logik Schicht
- Interfaces fachlich getrieben
- n:m Beziehung zu den Presentern

Implementierung

- Für die Demo `@ApplicationScoped`
- Stateless: State liegt in der UI Schicht

Copyright © 2013 – akquinet AG

Entkopplung durch Events

- CDI bietet einen Event Mechanismus
- Sender und Empfänger kennen sich nicht
- Events werden synchron verarbeitet
- Events werden zwischen Presentern ausgetauscht

Copyright © 2013 – akquinet AG

Wiederverwendung von Views/Presentern

- View bleibt unverändert
- Schicht mit Logik in Presenter einziehen
- Presenter und Events müssen qualifiziert werden
  - `@Qualifier`
  - Für jeden Anwendungsfall den korrekten Presenter
  - Auf die Events des korrekten Presenters reagieren

Copyright © 2013 – akquinet AG

Testen eines Presenters

- Die komplexe UI Logik liegt im Presenter
- Presenter können gut über Unittests getestet werden
- Junit, Mockito, Hamcrest
- Needle (`http://needle.spree.de`)

Copyright © 2013 – akquinet AG

Fazit und Fragen

- MVP ist ein bewährtes Pattern
- CDI bietet Kontext-abhängige Injection von Abhängigkeiten
- Über Events können Presenter entkoppelt werden
- Über Qualifier können Presenter wiederverwendet werden
- Gute Testbarkeit der Logik im Presenter

Copyright © 2013 – akquinet AG