5.– 8. September 2011
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Go Fullstack

Das Java-EE-Schichtenmodell für den Bau einer Webanwendung

Michael Kurz

irian | THE JAVA EXPERTS

Agenda

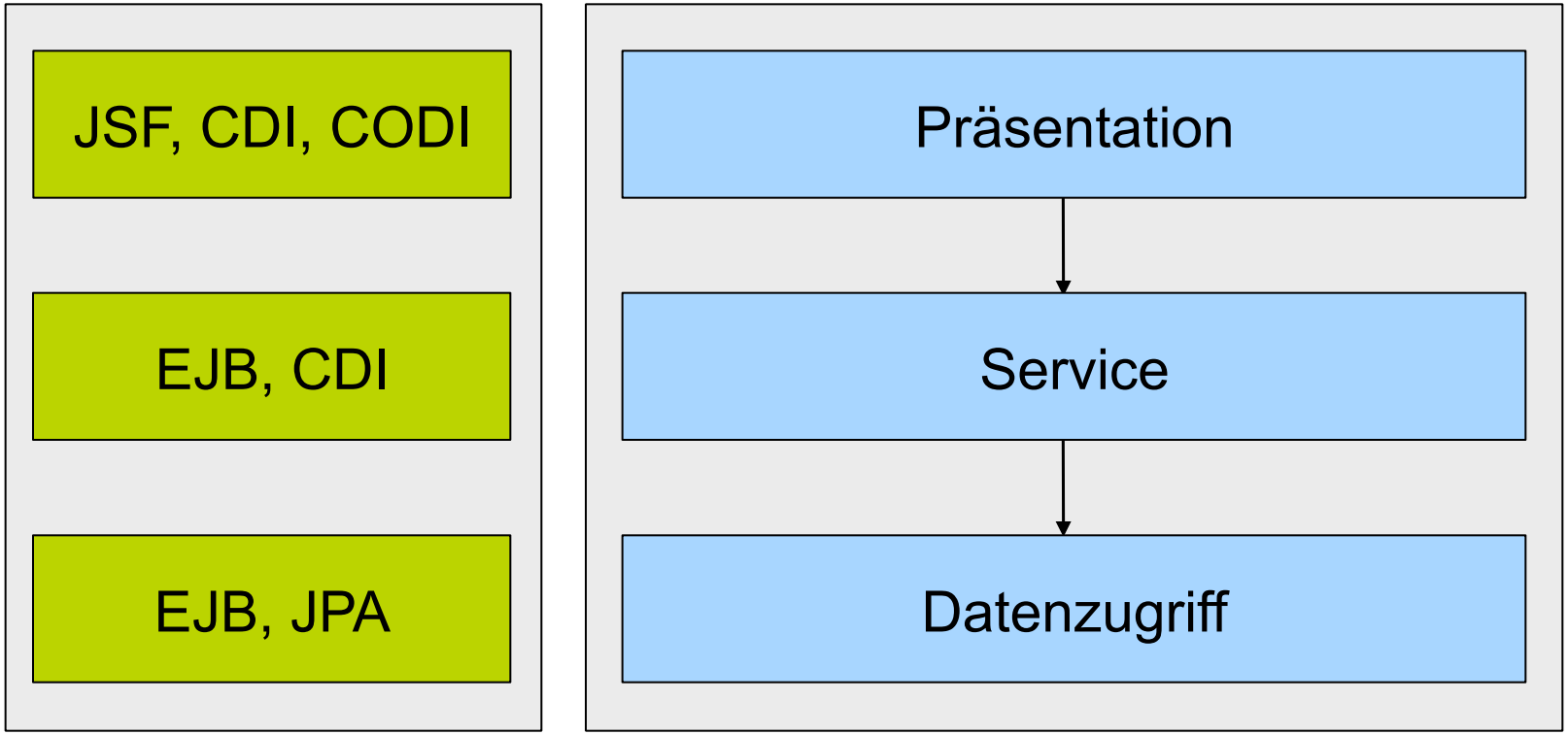
- Motivation
- Architektur und Technologie-Stack
- Details zu den einzelnen Schichten
- MyFaces CODI: CDI trifft JSF
- Demonstration anhand eines Beispiels
- MyGourmet EE (http://jsfatwork.irian.at)
- Keine Einführung in EJB, CDI, JSF...

Motivation

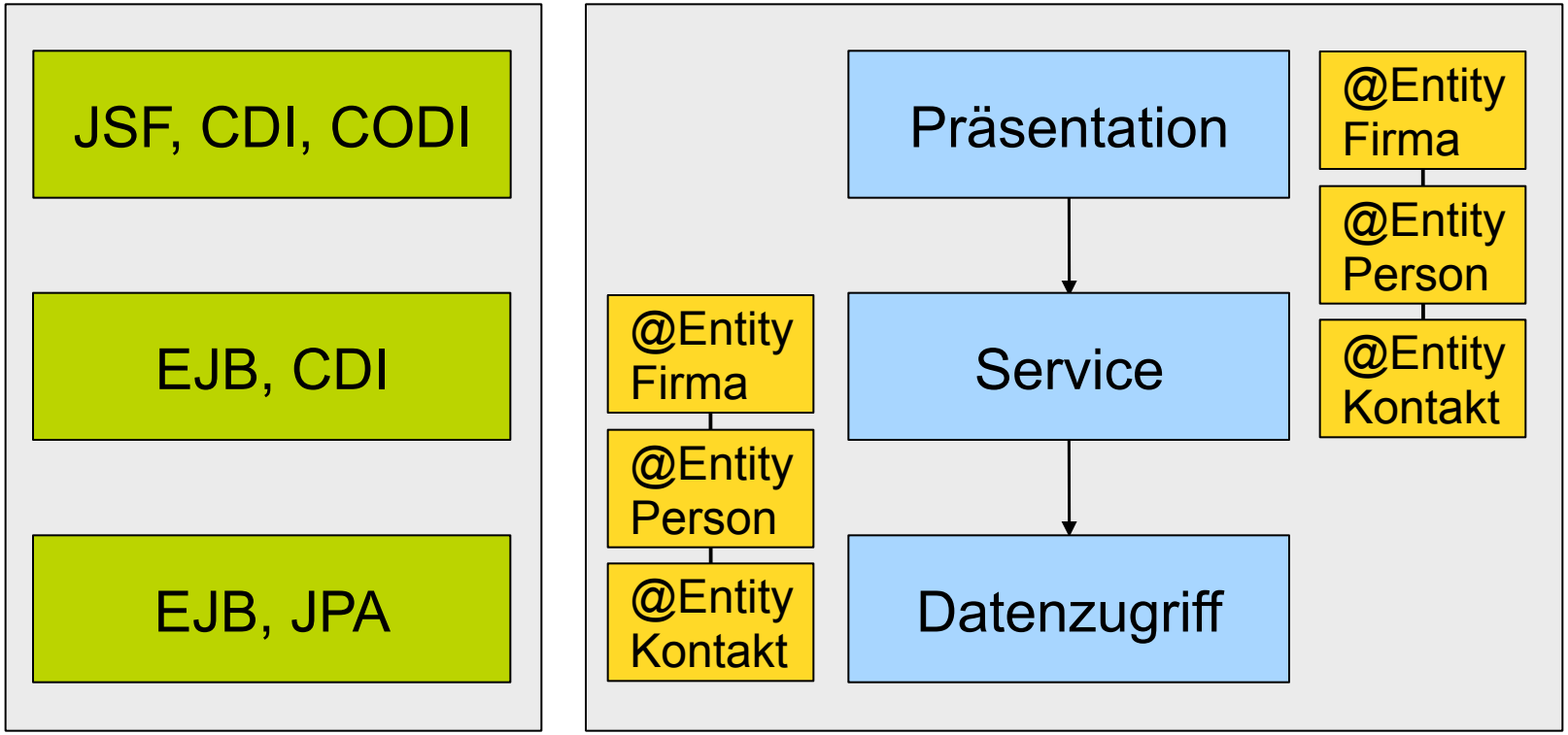
- Java EE 6 liegt im Trend
- Leichtigkeit des (Entwickler-)Seins
- Entwickler: Fokus auf Geschäftslogik
- Container: Services, Integration
- (Krasses) Beispiel aus der Praxis:
  - Alt: EJB 2.1 + Struts + MDA: 70000 loc
  - Neu: EJB 3.1 + CDI + JSF 2: 5500 loc

Der Vortragende warnt:
Dieser Vortrag kann zum Überdenken alter Muster und spontanen Neuimplementierungen führen!

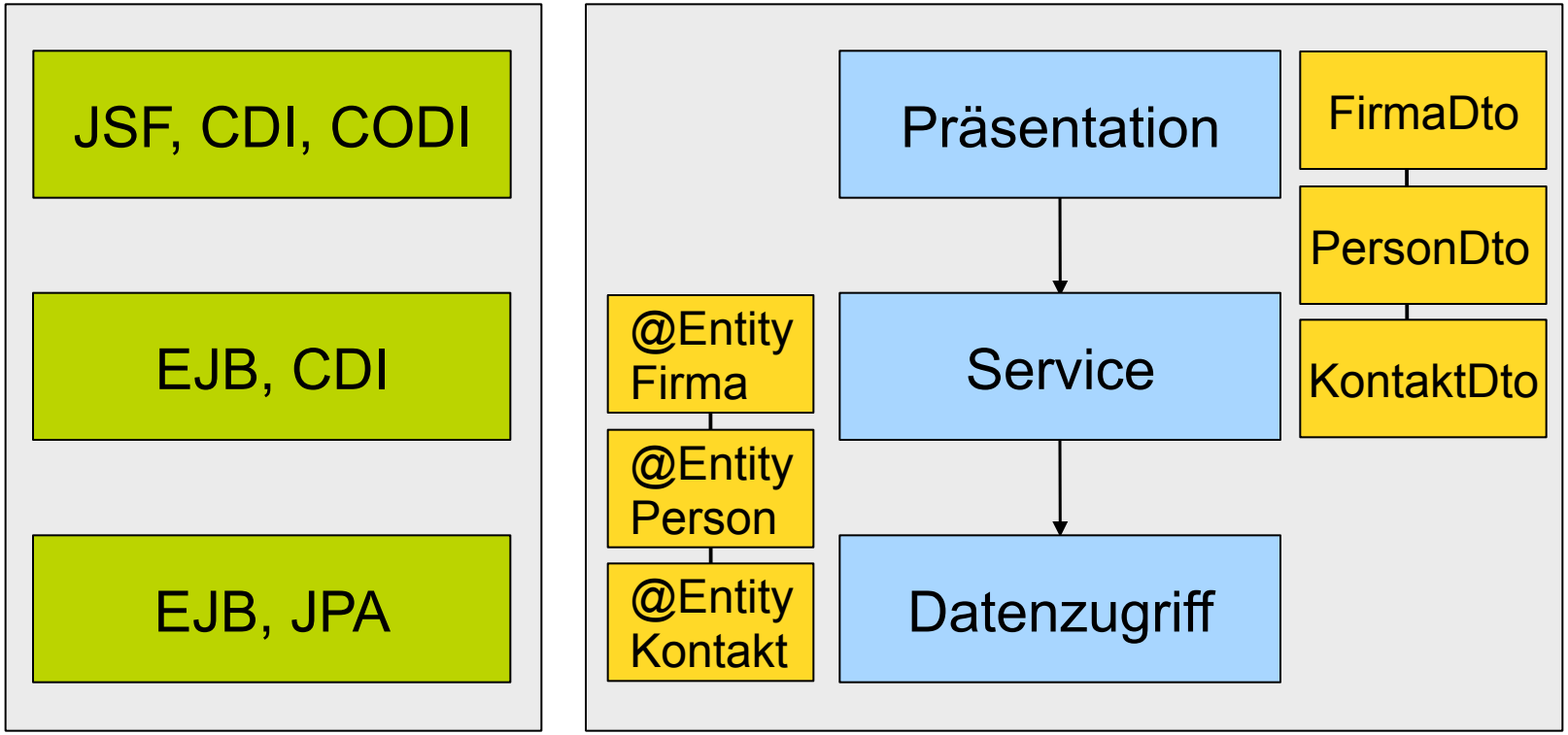
Architektur und Technologie-Stack



Architektur kleine/mittlere Projekte



Architektur große Projekte



Die Leichtigkeit von Java EE 6

- Führt Ease-of-Development-Strategie von Java EE 5 konsequent weiter
- EJB 3.1 auf Minimum reduziert:

```
@javax.ejb.Stateless
public class CustomerService {
  @javax.inject.Inject
  private CrudService crudService;
  ...
}
```

- CDI erhöht Flexibilität enorm
- WAR-Deployment von EJBs

Datenzugriffsschicht 1/2

- Kein DAO-Layer im klassischen Sinn
- Auf einen CRUD-Service reduziert
- Generischer Zugriff auf EntityManager
  - persist, merge, delete, find
- Ausführen von Queries
  - Named Queries
  - JPA 2 Criteria Queries
- Kann in komplexeren Applikationen Basis für weitere Services sein

Datenzugriffsschicht 2/2

- Service setzt Transaktion voraus

```
@javax.ejb.Stateless
@javax.ejb.TransactionAttribute(
    TransactionAttributeType.MANDATORY)
public class CrudService {
  @PersistenceContext
  private EntityManager em;
}
```

- In Serviceschicht eingeimpft (CDI)

```
@javax.inject.Inject
private CrudService crudService;
```

Serviceschicht 1/2

- Kapselt Geschäftslogik
- Service-Facade definiert Schnittstelle für Client
- Charakteristik Service-Facade:
  - Stateless-Session-Beans
  - Methoden starten Transaktion
  - Benutzen andere Services (CDI)
- Keine klassische Facade!
- Optional **@Remote** auf Service-Facade

Serviceschicht 2/2

- Beispiel einer Service-Facade:

```
@javax.ejb.Stateless
@javax.ejb.TransactionAttribute(
    TransactionAttributeType.REQUIRES_NEW)
public class CustomerService {
  @javax.inject.Inject
  private CrudService crudService;
}
```

- Transaktionen durch Container (default)

```
@javax.ejb.TransactionManagement(
  TransactionManagementType.CONTAINER)
```

Demonstration

Datenzugriffs- und Service-Layer

- Datenzugriffsschicht mit CRUD-Service
- Serviceschicht mit Service-Facades

Beispiel unter: **https://github.com/jsflive/mygourmet-confess2011**

Präsentationsschicht

- Webapplikation mit JSF 2.0
- JSF 2.0 vereinfacht Entwicklung enorm
- CDI verwaltet Managed-Beans

```
@javax.inject.Named
@javax.enterprise.context.RequestScoped
public class CustomerPage {
  @javax.inject.Inject
  private CustomerService service;
```

- Gut: JSF 2 + CDI
- Besser: JSF 2 + CDI + MyFaces CODI

MyFaces Extensions CDI

- Portable Erweiterung für CDI
- Vereinfacht Arbeit mit CDI und JSF
- Mehrere Module:
  - JSF 1.2 und 2.0
  - JPA, Bean-Validation, Message, Scripting...
- Funktioniert mit:
  - CDI: OpenWebBeans, Weld
  - JSF: MyFaces, Mojarra (1.2 und 2.0)
- Stabile Version 1.0.1 verfügbar

CODI: JSF Modul

- Erweitertes Konversationskonzept
  - **@ConversationScoped** (!= CDI)
  - **@WindowScoped**
  - **@ViewAccessScoped**
- Ermöglicht CDI mit JSF-Annotationen
- **@Inject** für folgende JSF-Artefakte:
  - Konverter, Validatoren, Phase-Listener
- Typsichere View-Config, Page-Beans
- **@Inject** für Constraint-Validator (BV)

CODI: Konversationen 1/3

- Mehrere Konversationen pro Ansicht
- Konversation an Tab/Fenster gebunden
- Konversation beginnt mit Bean-Zugriff
- Manuell oder durch Timeout beendet

```
@Named @ConversationScoped
public class WizardBean {
  @Inject
  private Conversation conversation;
  public void finish() {
    this.conversation.close();
  }
```

CODI: Konversationen 2/3

- Standard: Eine Konversation pro Bean
- **@ConversationGroup** um mehrere Beans in Konversation zu gruppieren

```
public interface Wizard {}
@Named @ConversationScoped
@ConversationGroup(Wizard.class)
public class Page1 {}
@Named @ConversationScoped
@ConversationGroup(Wizard.class)
public class Page2 {}
```

CODI: Konversationen 3/3

- **@ViewAccessScoped**
  - Beginnt mit erstem Bean-Zugriff
  - Lebensdauer pro Ansicht erweitert
  - Bis Bean in Ansicht nicht benutzt wird
- **@WindowScoped**
  - Konversation pro Fenster/Tab
- Zugriff auf WindowContext möglich

```
@Inject
private WindowContext windowContext;
```

Demonstration

MyFaces CODI: Konversationen

Beispiel unter: **https://github.com/jsflive/mygourmet-confess2011**

CODI: @Inject für JSF-Artefakte

- **@Advanced** ermöglicht **@Inject** für:
  - Konverter
  - Validatoren
  - Phase-Listener

```
@Advanced
@FacesConverter("categoryConverter")
public class CategoryConverter {
  @Inject
  private FinderService finderService;
  ...
}
```

CODI: View-Konfiguration 1/2

- View-Konfiguration für /myPage.xhtml

```
@Page
@PageBean(MyPageBean.class)
public final class MyPage implements ViewConfig {}
```

- Metadaten wie Page-Beans
- Typsichere Navigation

```
public Class<? extends ViewConfig> next() {
  return MyPage.class;
}
```

CODI: View-Konfiguration 2/2

- Page-Beans als Event-Listener

```
public class MyPageBean {
  @InitView
  protected void initView() {}
  @PrePageAction
  protected void prePageAction() {}
  @PreRenderView
  protected void preRenderView() {}
  @BeforePhase(PROCESS_VALIDATIONS)
  protected void preValidate() {}
  @AfterPhase(PROCESS_VALIDATIONS)
  protected void postValidate() {}
}
```

Demonstration

MyFaces CODI im Einsatz

- **@Inject** in JSF-Artefakten
- View-Konfiguration und Page-Beans

Beispiel unter: **https://github.com/jsflive/mygourmet-confess2011**

Fazit

- JavaEE 6 erlaubt schlanke Architektur
- Starke Konkurrenz zu Spring
- Alle Schichten „standardisiert“
- Portabilität von Code und Wissen
- JSF 2.0 + CDI + MyFaces CODI hebt UI-Entwicklung auf neue Ebene

Vielen Dank!

Michael Kurz

Irian Solutions GmbH

Neugierig?

- Marinschek, Kurz, Müllan: **JavaServer Faces 2.0**, dpunkt.Verlag
- Irian JSF@Work Online-Tutorial **http://jsfatwork.irian.at**
- JSFlive Weblog **http://jsflive.wordpress.com**

JSF live