

3.– 6. September 2012
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

Schlankheitskur

Lean Web Architecture mit JSF 2.0, CDI & Co.

Andreas Hartmann

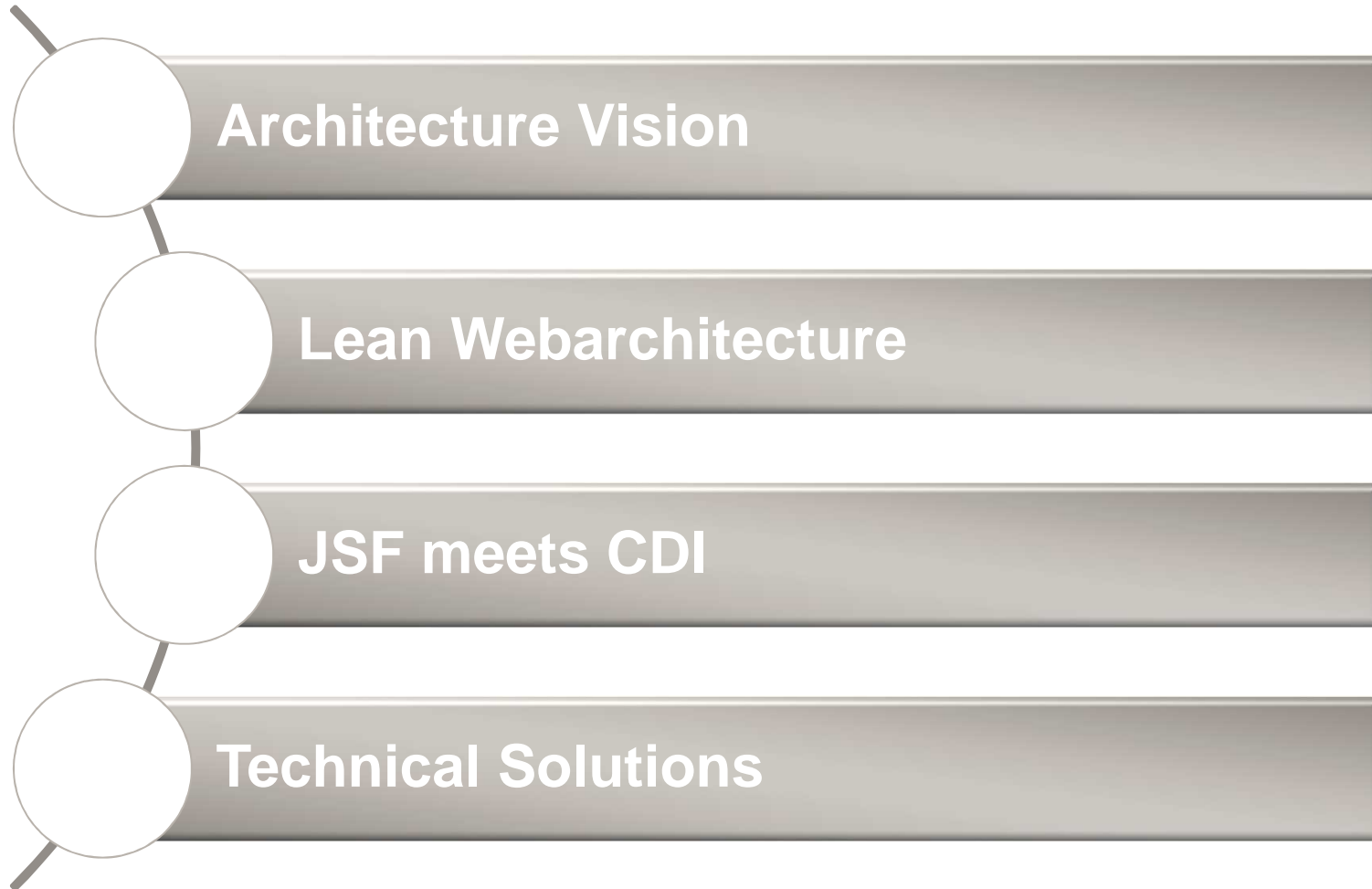
adesso AG

Schlankheitskur

Lean Web Architecture mit JSF 2.0, CDI & Co.

Andreas Hartmann

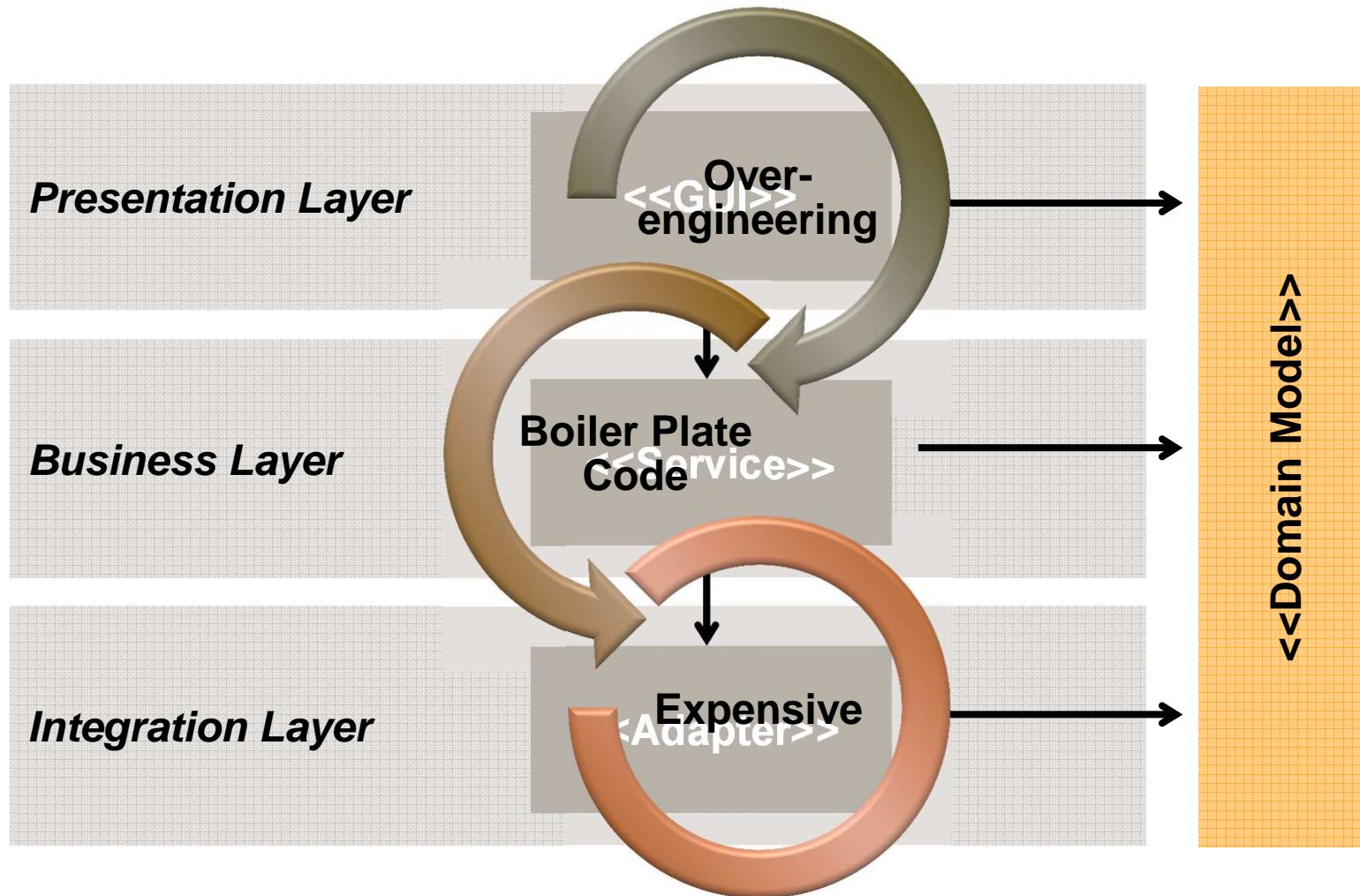
05.09.2012



Architecture Vision



Architecture Vision – 3 Layer Reflex

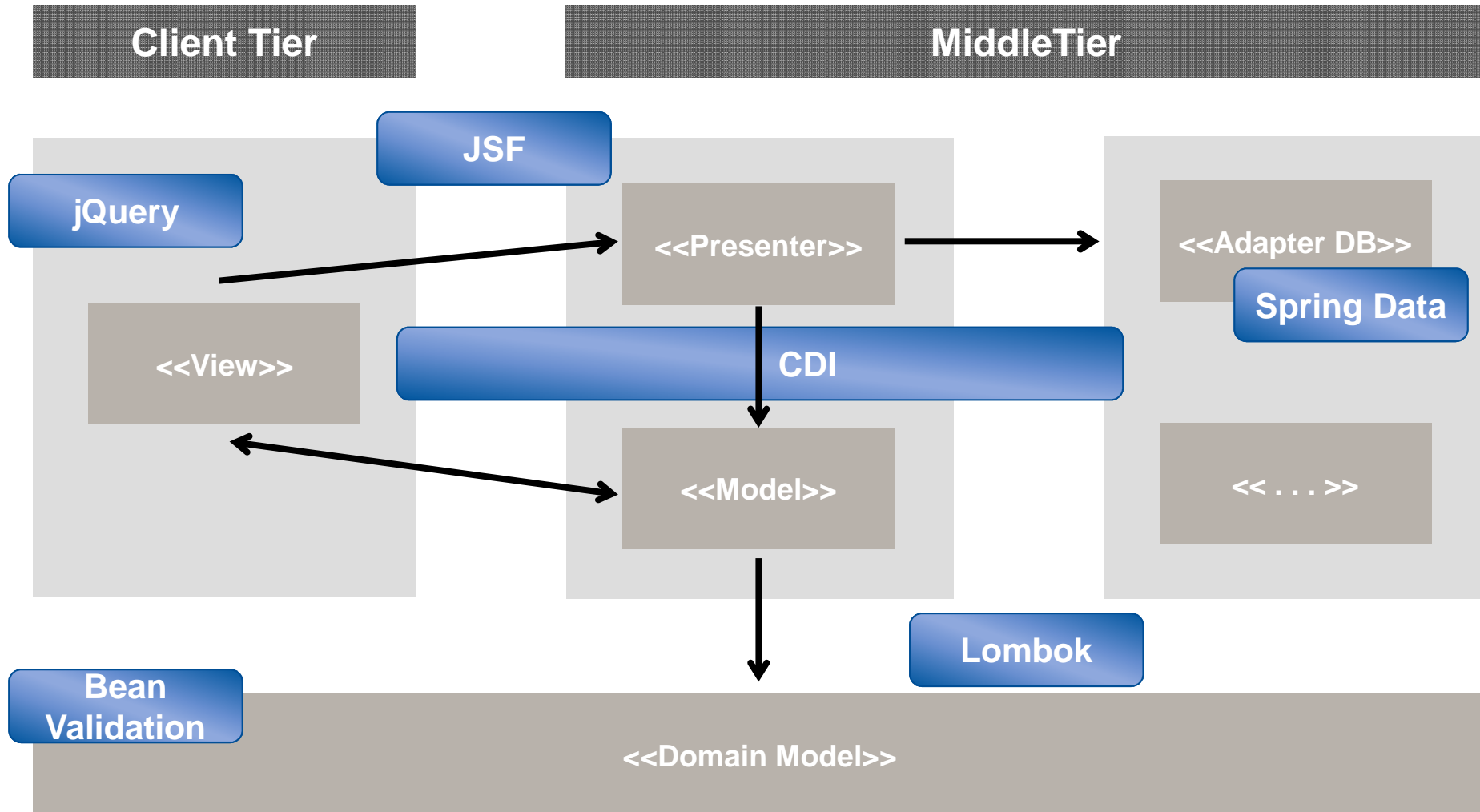


Transaction-Handling
Exception-Handling
Logging
Performance-Monitoring
Security
Composite-Components
Configuration

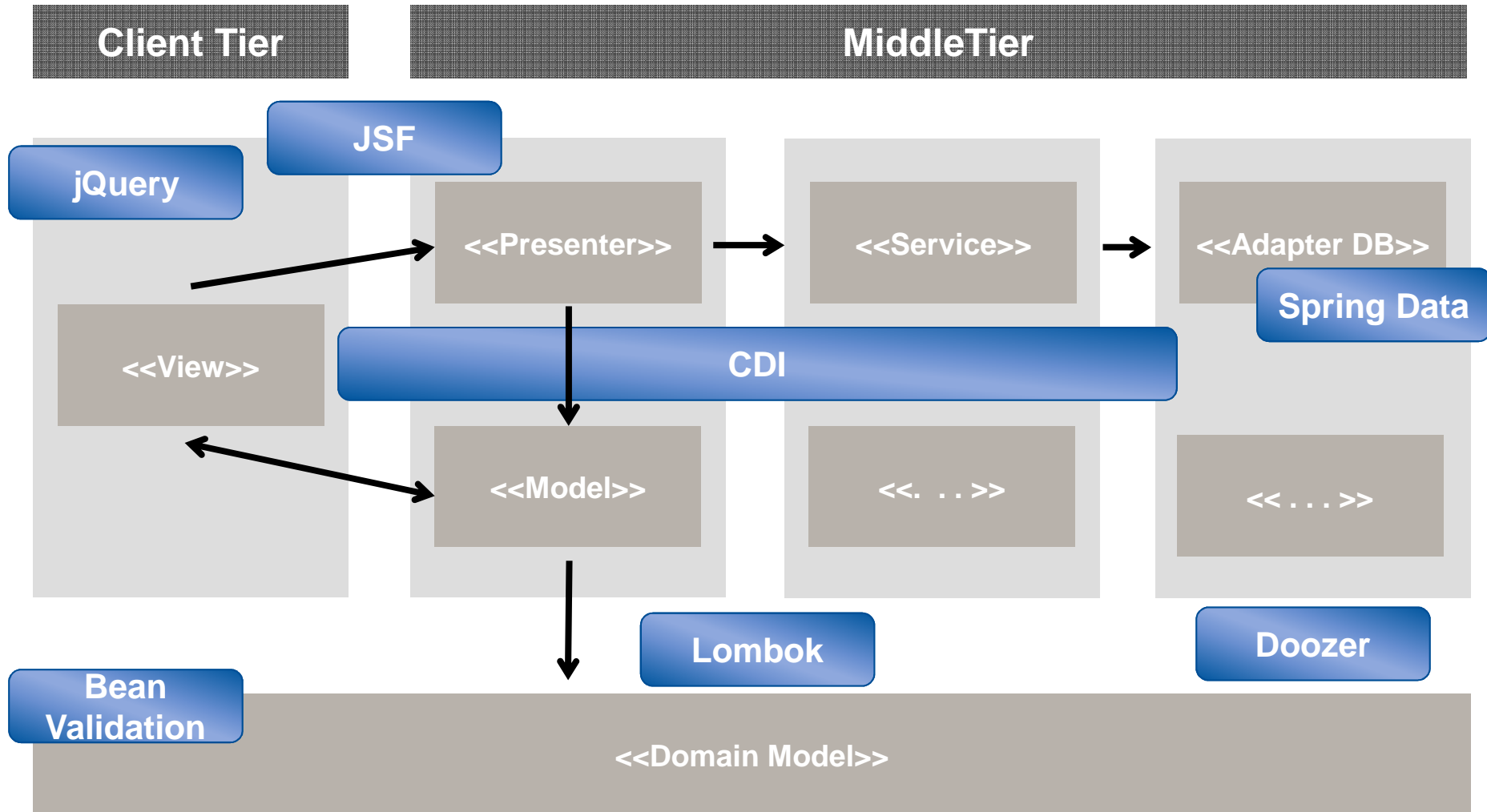
Architecture Vision – Technology Stack



Lean Webarchitecture – Tiny Applications



Lean Webarchitecture – Large Scale Applications



Dependency Injection

- ▶ Container steuert Objekte und Abhängigkeiten
 - > Instanziiert Objekte
 - > Injiziert Abhängigkeiten zur Laufzeit (lose Kopplung)
- ▶ Konfiguration über Annotations oder XML-Datei
- ▶ Populär geworden durch das Spring Framework
- ▶ Seit EJB3.0 auch in Java EE verfügbar
 - > @EJB
 - > @PersistenceUnit
 - > @Resource



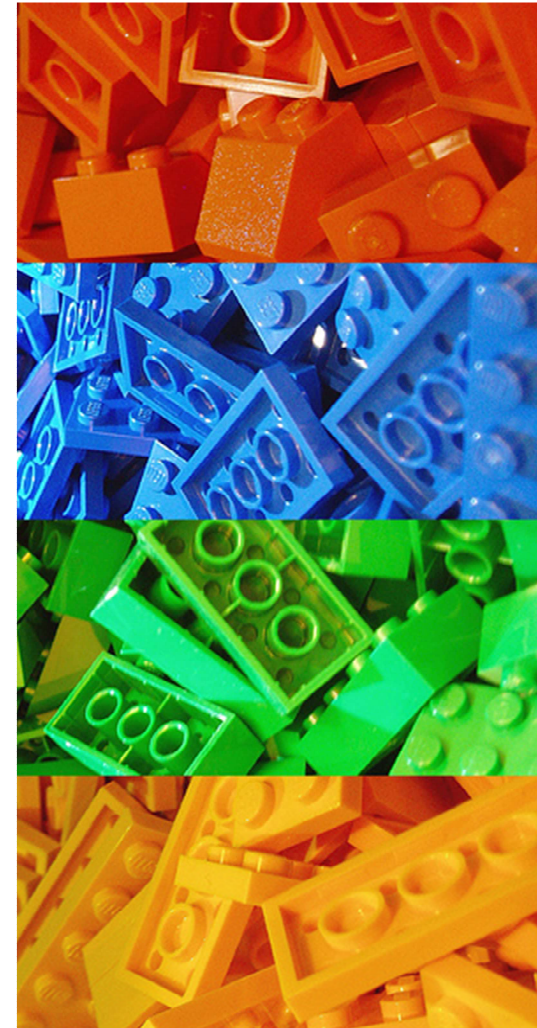
Dependency Injection mit CDI

- ▶ Standardisiertes DI für Java (SE/EE)
 - > Typsicherheit
 - > Wiederverwendbarkeit
 - > Testbarkeit
 - > Flexibilität
 - > Lifecycle „Awareness“
- ▶ User Story driven injizieren

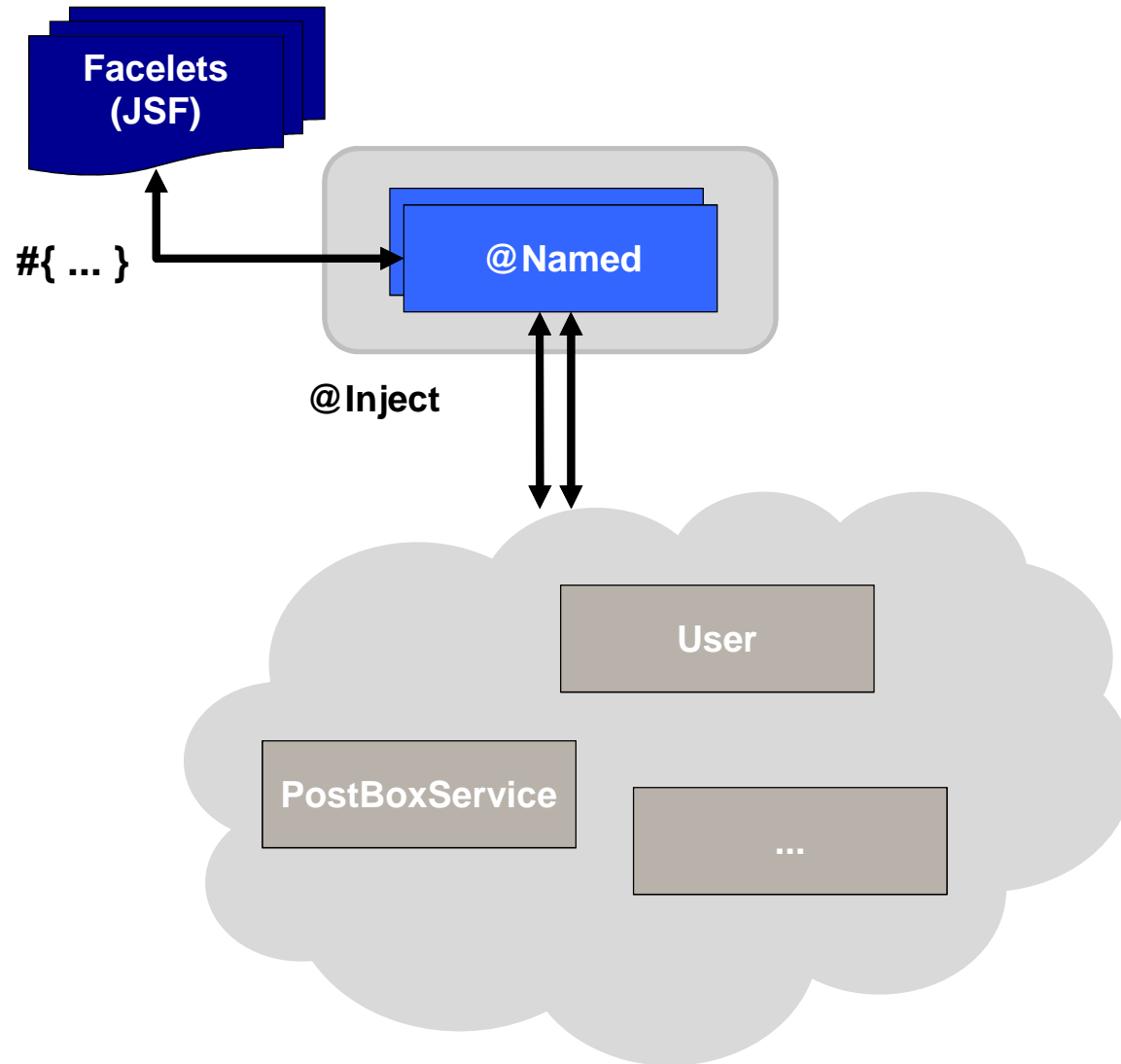


CDI Bausteine

- ▶ Typsicherheit
 - > Stereotypes
 - > Qualifier
 - > Alternatives
- ▶ Lose Kopplung
 - > Interceptors
 - > Decorator
 - > Events
- ▶ Sichtbarkeiten
 - > Scopes
 - > Lifecycle Awareness



CDI im Detail



CDI Managed Bean

```
import javax.enterprise.context.RequestScoped;  
import javax.inject.Named;  
import javax.inject.Inject;
```

```
@Named("userController")  
@RequestScoped  
public class UserController {
```

```
    @Inject  
    private UserService userService;
```

```
    private List<User> users;
```

```
    @PostConstruct  
    private void init() {  
        this.users = userService.findAllUsers();  
    }  
}
```

#{userController}

gültig für Request

POJO

Injection Point

CDI Injection Points

- ▶ Field Injection

```
@Inject  
private UserService userService;
```

- ▶ Constructor Injection

```
@Inject  
public UserController(UserService userService) {  
    this.userService = userService;  
}
```

- ▶ Setter Injection

```
@Inject  
public void setUserService(UserService userService) {  
    this.userService = userService;  
}
```

Qualifier & Producer

- ▶ Aktuellen Benutzer injizieren

```
@Inject @Current private User user;
```

- ▶ Fachlichkeit injizieren

```
@Inject @Admin private List<User> administrators;
```

```
@Inject @Editor private List<User> editors;
```

- ▶ Infrastruktur injizieren

```
@Inject @UserDB private EntityManager userDB;
```

```
@Inject @ProductDB private EntityManager productDB;
```


User Story driven injizieren

Qualifier & Producer

```
import de.adesso.cdi.common.Current;
```

```
import javax.enterprise.inject.Produces;
```

```
@Named
```

```
@SessionScoped
```

```
public class Authenticationimplements Serializable {
```

```
    private User authenticatedUser;
```

```
    public String authenticate() {...}
```

```
@Produces
```

```
@Named("authenticatedUser")
```

```
@Current
```

```
public User getAuthenticatedUser() {
```

```
    return authenticatedUser;
```

```
}
```

EL: #{authenticatedUser}

Fachlichkeit statt Infrastruktur injizieren

Qualifier & Producer

```
import de.adesso.cdi.common.Current;
```

```
public class CustomerService implements CustomerService {
```

```
    @Inject @Current
```

```
    private User currentUser;
```

```
    public void addCustomer(Customer customer) {  
        customer.setCreator(currentUser);  
        em.persist(customer);  
    }
```

HTML 5

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml111/"
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:f="http://java.sun.com/jsf/core">
<h:head>
  <title><ui:insert name="pageTitle"></ui:insert></title>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
  <h:outputScript name="js/jquery.js" />
  <h:outputScript name="js/jquery-ui.min.js" />
  <h:outputScript name="js/jquery-ui.datepicker-de.js" />
  <h:outputScript name="js/fileuploader.js" />
  <h:outputScript name="js/ui.multiselect.js" />
  <h:outputScript name="js/jquery.dataTables.min.js" />
  <h:outputStylesheet name="css/reset.css" />
  <h:outputStylesheet name="css/jquery-ui.css" />
  <h:outputStylesheet name="css/ui.multiselect.css" />
  <h:outputStylesheet name="css/style.css" />
</h:head>
```

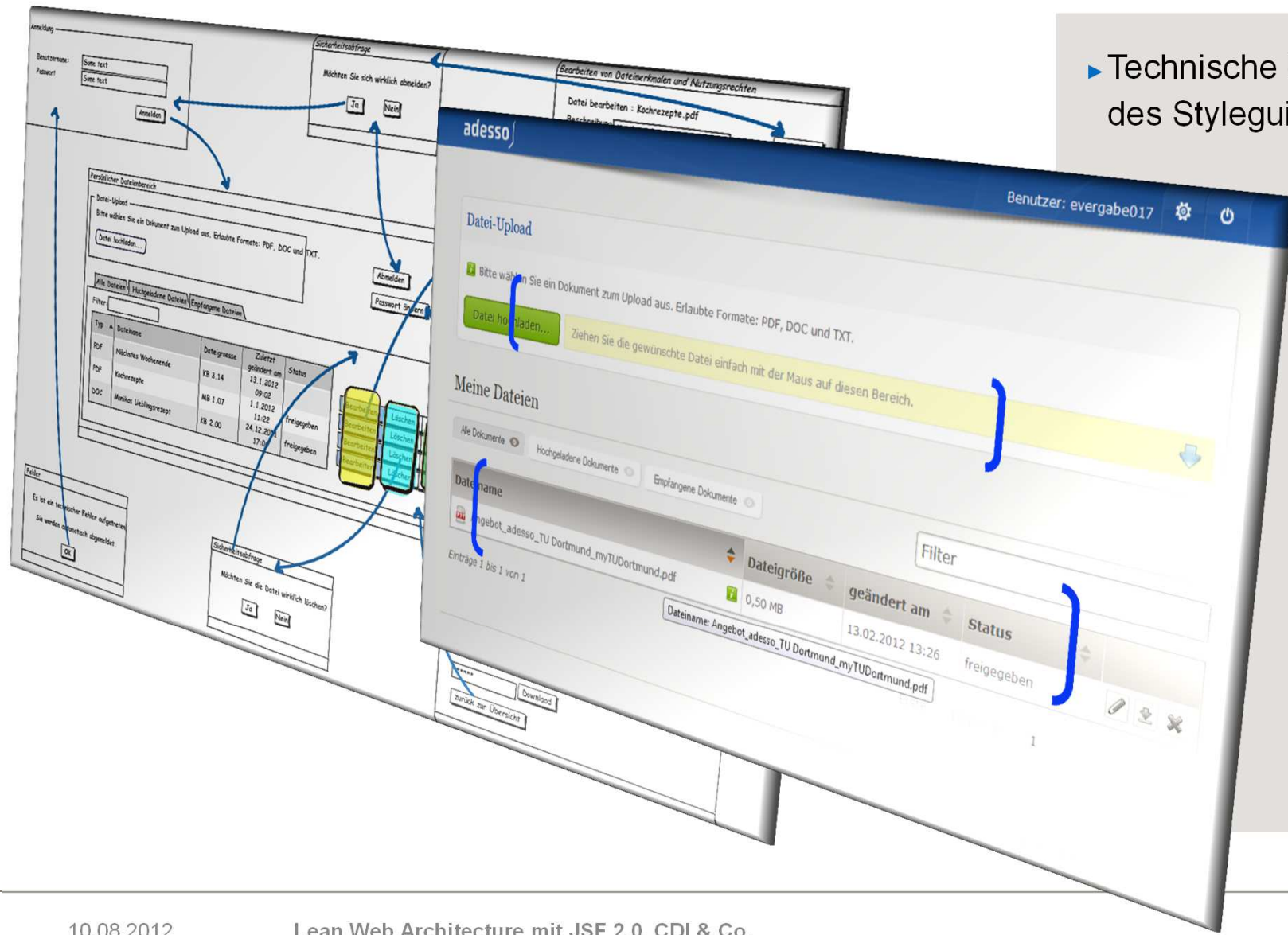
- ▶ Mit JSF2.2 auch HTML5 Komponenten



Custom Components

10550
prof. technolo

► Technische Umsetzung des Styleguides



```
<script type="text/javascript">
/*  */
jQuery(document).ready( function() {
    var browserIE8orLess = $.browser.msie &amp;&amp; $.browser.version &lt;= 8;
    if (!browserIE8orLess) {
        var table=$('#documentForm');
    }
    &lt;h:dataTable id="documentTable"
        rendered="#{not empty fileExchangePresenter.model.documents}"
        rowClasses="odd,even"
        value="#{fileExchangePresenter.model.documents}" var="document"
        styleClass="data-table"&gt;
    &lt;h:column headerClass="fileNameCol"&gt;
        &lt;f:facet name="header"&gt;
            &lt;h:outputText value="#{msg.fileName}" /&gt;
        &lt;/f:facet&gt;
        &lt;h:outputText value="#{document.fileName}" styleClass="icon" /&gt;
        &lt;h:graphicImage rendered="#{not empty document.description}"
            url="/resources/images/icon_info.png"
            alt="Beschreibung: #{document.description}"
            title="Beschreibung: #{document.description}"
            styleClass="description" /&gt;
        &lt;/h:column&gt;
    &lt;h:column headerClass="fileSizeCol"&gt;</pre></div><div data-bbox="681 222 912 514" data-label="List-Group"><ul><li>▶ freie, umfangreiche JavaScript Bibliothek</li><li>▶ komfortable Möglichkeit zur DOM Manipulation</li><li>▶ Erweiterbarkeit durch Plugins (u.a. DataTable)</li></ul></div><div data-bbox="730 763 890 819" data-label="Image"><img alt="jQuery logo with the tagline 'write less, do more.'"/></div><div data-bbox="117 907 183 929" data-label="Page-Footer"><p>10.08.2012</p></div><div data-bbox="242 907 506 930" data-label="Page-Footer"><p>Lean Web Architecture mit JSF 2.0, CDI &amp; Co.</p></div><div data-bbox="835 900 910 936" data-label="Page-Footer"><p>adesso</p></div>
```

Lombok

```
public class Email implements Serializable {  
    @Getter @Setter  
    private Object id;  
  
    @NotNull  
    @Getter @Setter  
    private Date incomingDate;  
  
    @NotNull  
    @Getter @Setter  
    private String subject;  
  
    @Getter @Setter  
    private String text;  
}
```

- ▶ Reduziert Boilerplate Code durch Annotations
 - > @Getter, @Setter
 - > @ToString
 - > @HashCodeAndEquals
 - > @Data
 - > ...
- ▶ Delombok Funktionalität
- ▶ Eclipse Support



JSR 303: Bean Validation

```
@Entity
@Table(name = "USERS")
public class User {

    /**
     * Loginname des Benutzers.
     */
    @Id
    @Column(name = "IAM_USERNAME", unique = true)
    @Length(max = 15)
    private String username;

    /**
     * Passwort des Benutzers.
     */
    @Column(name = "IAM_PASSWORD")
    @NotNull
    @Length(max = 255)
    private String password;
}
```

```
graph TD
    subgraph Java
        Client[Client]
        PL[Presentation Layer]
        BL[Business Layer]
        DAL[Data Access Layer]
        DB[(Database / Disk)]
        Client --> PL
        PL --> BL
        BL --> DAL
        DAL --> DB
    end
    DM[Domain Model] --> Client
    DM --> PL
    DM --> BL
    DM --> DAL
```

- ▶ Standardisierung von Constraints in Form von Annotations
- ▶ Standardisierung des Metadata API
- ▶ Integration anderer Frameworks wie z.B. JSF und JPA

Doozer

Lesson / People technology

- ▶ Java Bean to Java Bean Mapper

```
<?xml version="1.0" encoding="UTF-8"?>  
<Mappings xmlns="http://dozer.sourceforge.net"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://dozer.sourceforge.net  
    http://dozer.sourceforge.net/schema/beanm
```

```
<mapping>  
  <class-a>de. adesso.mvc4jsf.showcase.domain.EmailEntity</a>  
  <class-b>de. adesso.mvc4jsf.showcase.adapter.entity.EmailEntity</b>  
  <field custom-converter="de. adesso.mvc4jsf.showcase.adapter.entity.converter.DateConverter">  
    <a>id</a>  
    <b>id</b>  
  </field>  
  <field>  
    <a>incomingDate</a>  
    <b>incomingDate</b>  
  </field>  
  <field>  
    <a>subject</a>  
    <b>subject</b>  
  </field>  
  <field>  
    <a>text</a>
```

```
public class Email implements Serializable {  
  @Getter @Setter  
  private Object id;  
  
  @NotNull  
  @Getter @Setter  
  private Date incomingDate;  
  
  @NotNull  
  @Getter @Setter  
  private String subject;  
  
  @Getter @Setter  
  private String text;  
}
```

```
@ApplicationScoped  
@Named(value="userDAO")  
public class UserDAO implements Serializable {  
  @Inject DomainModelMapper modelMapper;  
  @Inject DataSource dataSource;  
  
  public boolean deleteEmail(Email email) {  
    EmailEntity emailEntity = modelMapper.mapToMorphia(email);  
    dataSource.getDataStore().delete(emailEntity);  
    return true;  
  }  
  
  public Email saveEmail(Email email) {  
    EmailEntity emailEntity = modelMapper.mapToMorphia(email);  
    dataSource.getDataStore().save(emailEntity);  
    return modelMapper.mapToDomainModel(emailEntity);  
  }  
}
```



SpringData

```
public interface UserRepository extends
    PagingAndSortingRepository<User, String> {
    /**
     * Liefert alle User außer den angegebenen, sortiert nach Name.
     *
     * @param blacklist Liste der User
     * @return Liste der User.
     */
    @Query("SELECT u FROM User u WHERE u NOT IN (:blacklist)")
    List<User> findAllOrderByUsernameE

    count(): long - Override method in 'CrudRepo
    delete(Iterable<? extends User> arg0): void -
    delete(String arg0): void - Override method in
    delete(User arg0): void - Override method in
    deleteAll(): void - Override method in
    exists(String arg0): boolean - Override method

    @Entity
    @Table(name = "USERS")
    public class User {
        /**
         * Loginname des Benutzers.
         */
        @Id
        @Column(name = "IAM_USERNAME", unique = true)
        @Length(max = 15)
        private String username;

        /**
         * Passwort des Benutzers.
         */
        @Column(name = "IAM_USER_PWD")
        @NotNull
        @Length(max = 255)
        private String password;
    }
}
```

- ▶ ermöglicht Zugriff auf relationale und NoSQL Datenbanken
- ▶ CDI Integration





3.– 6. September 2012
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

Vielen Dank!

Andreas Hartmann

adesso AG