5.– 8. September 2011
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

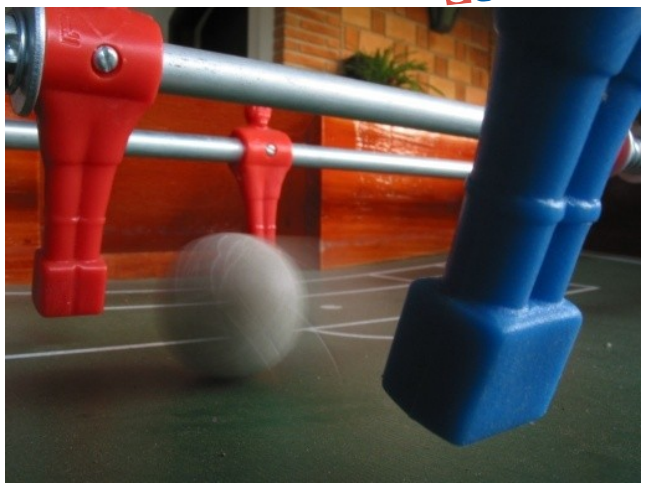
Türsteher für Bohnen

Bean Validation mit JSR-303

Simon Zambrovski

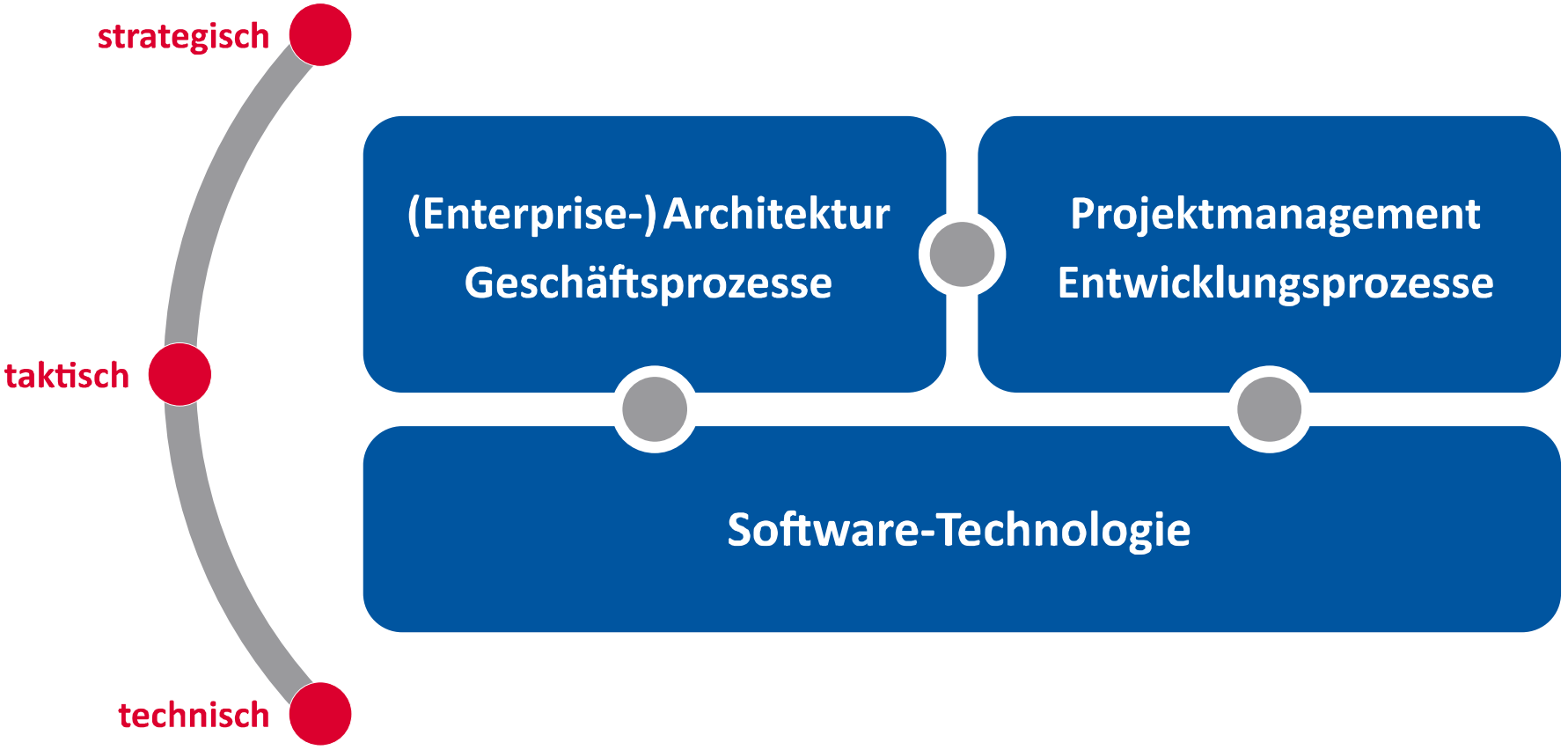
Holisticon AG

Holisticon AG



- Holisticon wurde 2006 gegründet.
- Das operative Geschäft in Deutschland haben wir zum 1. Januar 2007 aufgenommen.
- Wir sind in Großraum Hamburg vertreten.
- Wir beraten Konzerne und große mittelständische Unternehmen.
- Wir sind ein Tochterunternehmen der schwedischen Sigma Gruppe.

Unser Weltbild



Agenda

- Validierung in üblichen Architekturen
- JSR-303
- Domänenspezifische Datentypen
- Integration
  - Spring MVC
  - JSF2
  - JFace Data Binding
- Service Boundary Validation
- Offene Punkte

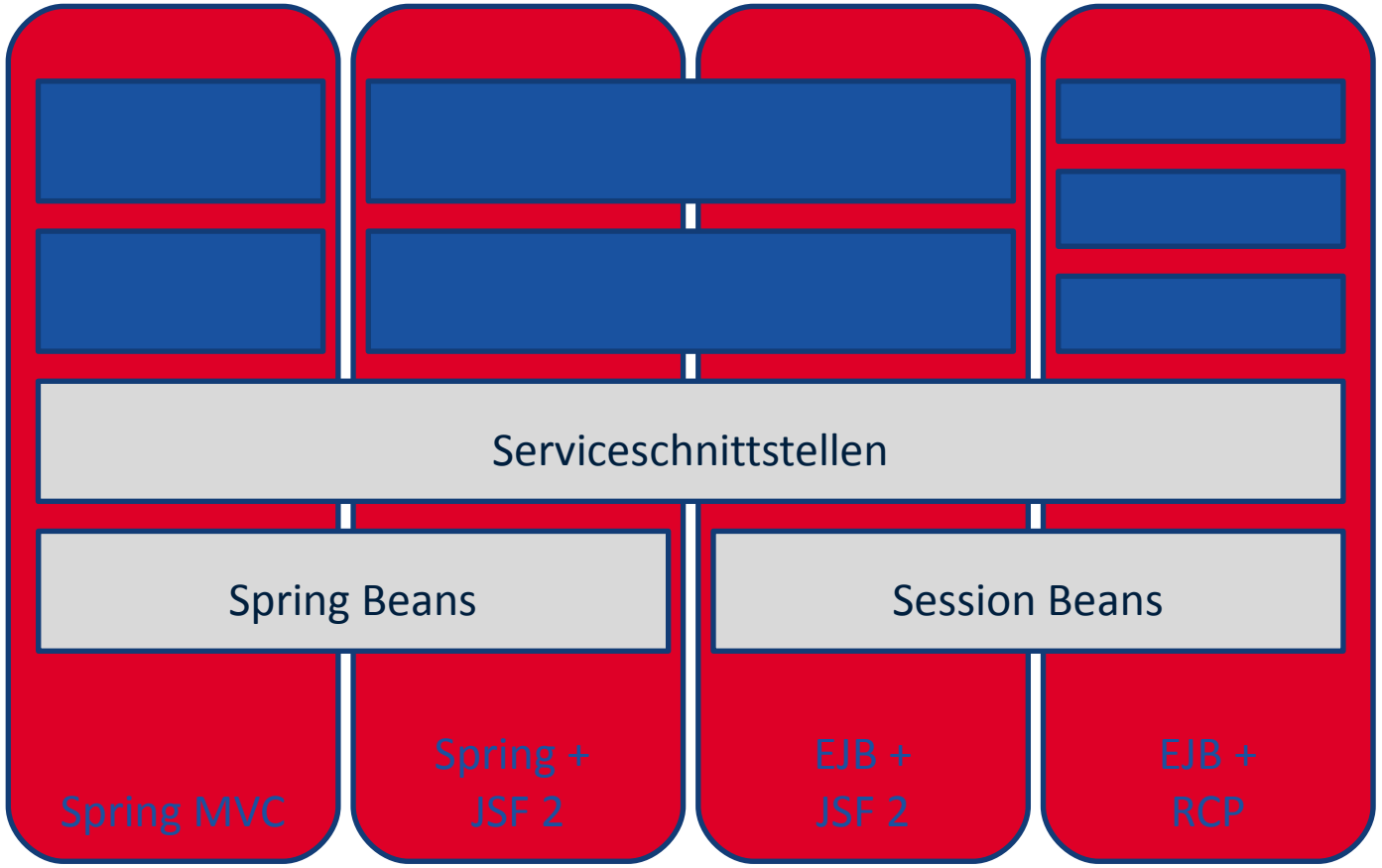
holisticon
MEMBER OF SIGMA GROUP

Validierung in üblichen Architekturen

Übliche Architekturen

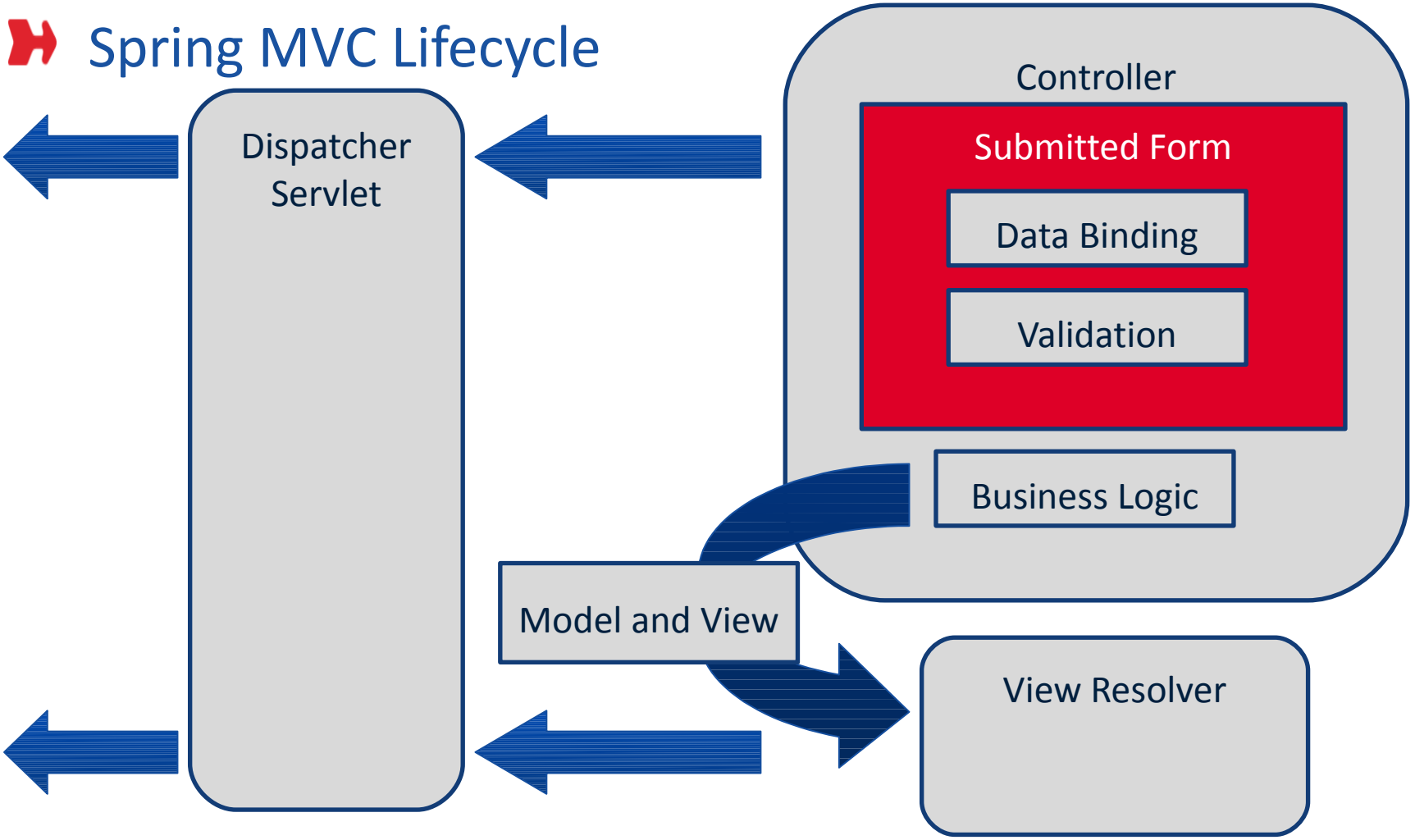
- Spezifika der Java Enterprise Welt
  - Mehrschicht-Architekturen
  - Bit-Schieber-Anwendung
  - Benutzer-Interaktion
  - Verwendung von (vielen verschiedenen) Frameworks
- Validierung
  - Eingabevalidierung
  - Design-by-Contract
  - Daten-Integrität

Übliche Architekturen



Übliche Architekturen

Spring MVC Lifecycle



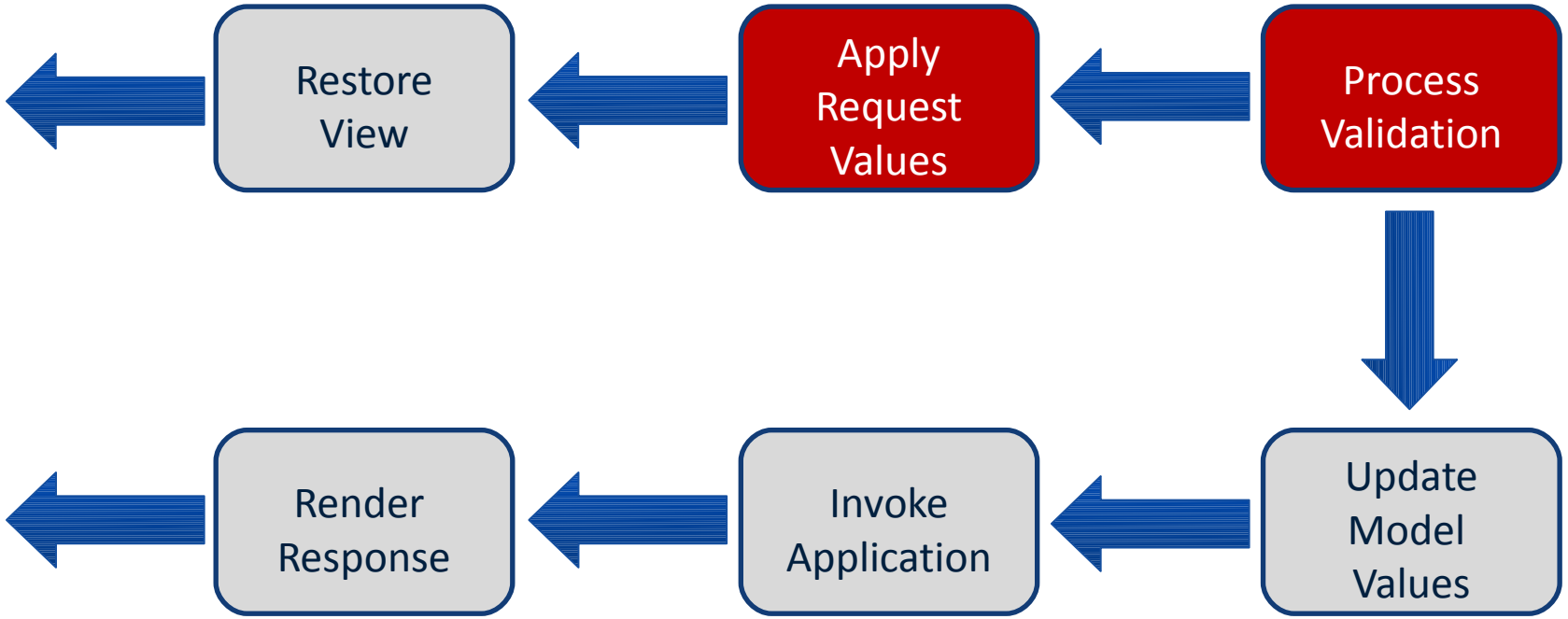
Übliche Architekturen

Spring MVC Validatoren (klassischer Ansatz)

```
public class ZipCodeValidator implements Validator {

  boolean supports(Class clazz) {
    return String.class.equals(clazz);
  }

  public void validate(Object obj, Errors e) {

    ValidationUtils.rejectIfEmpty(e, “zipcode", “zipcode.empty");
    String z = (String) obj;

    if (z.length() != 5) {
      e.rejectValue(”zipcode", ”wrong.size”);
    }
  }
}
```

Übliche Architekturen

JSF2 Validatoren (klassischer Ansatz)



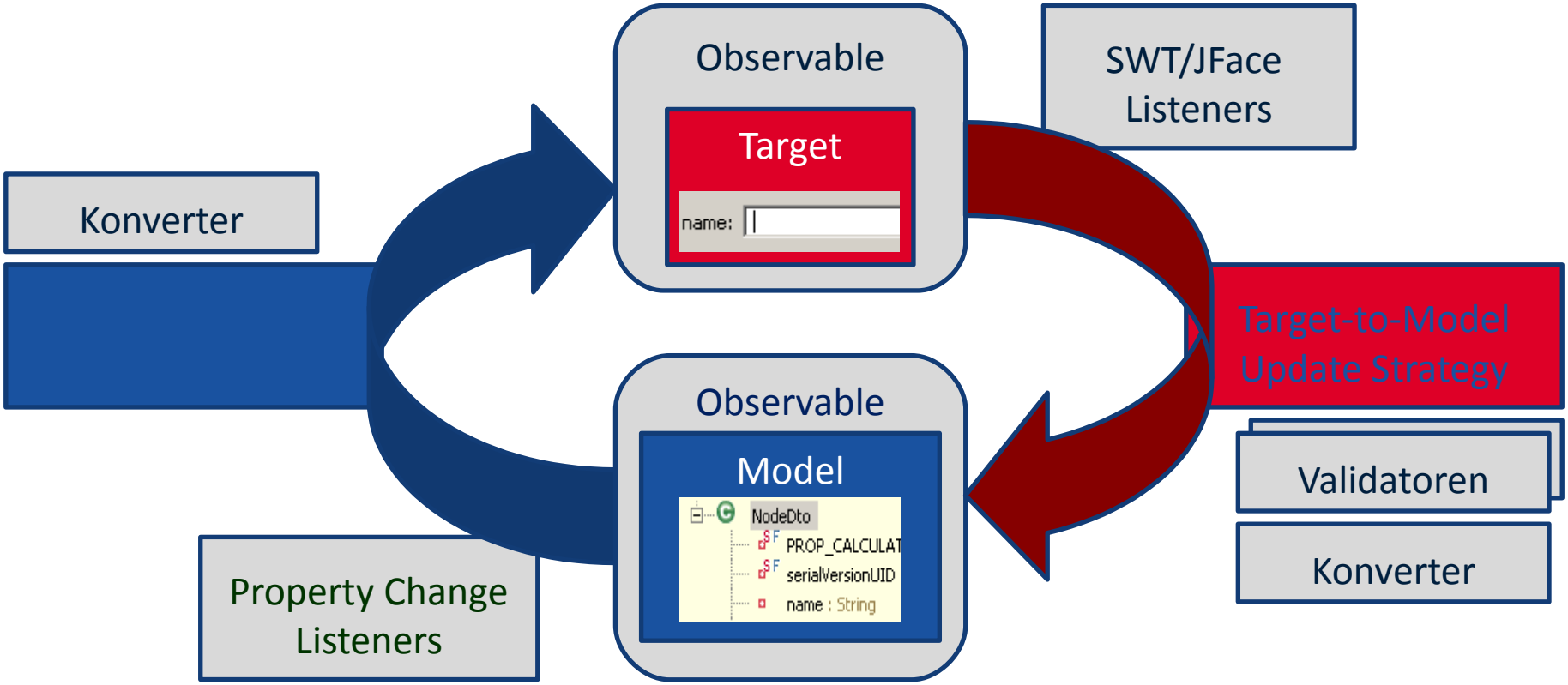
Übliche Architekturen

JSF2 Validatoren (klassischer Ansatz)

```
<h:inputText … >
  <f:validator validatorId="ZipcodeValidator" />
</h:inputText>

@FacesValidator("ZipcodeValidator")
public class ZipcodeValidator implements Validator {

  public void validate(FacesContext ctx, UIComponent component,
   Object value) throws ValidatorException {

    if (value instanceof String) {
      String zipCode = (String) value;
      if (zipCode == null || zipCode.length() != 5) {
         throw new ValidatorException(new FacesMessage(
         FacesMessage.SEVERITY_ERROR, "Wrong zip code", null));
      }
    }
  }
}
```

Übliche Architekturen

JFace Data Binding Validatoren (klassischer Ansatz)



Übliche Architekturen

JFace Data Binding Validatoren (klassischer Ansatz)

```
public class ZipCodeComposite extends Composite {

  public void createBinding(DataBindingContext dbc) {
    IObservableValue observableModel = …
    ISWTObservableValue observableTarget = …

    UpdateValueStrategy t2m = new UpdateValueStrategy();
    t2m.setAfterConvertValidator(new IValidator() {

      public IStatus validate(Object value) {
        if ((String) value == null || ((String) value).length() != 5) {
          return ValidationStatus.error("Wrong zip code");
        }
        return ValidationStatus.ok();
      }
    });

    dbc.bindValue(observableTarget, observableModel, t2m, null);
  }
}
```

Übliche Architekturen

JPA Hooks (klassischer Ansatz)

```
@ExcludeDefaultInterceptors
public class PersistenceValidationListener {

  @PrePersist
  @PreUpdate
  @PreRemove
  public void validateMe(Object entity) {

    if (entity instanceof Address) {
      String zipCode = ((Address)entity).getZip();
      if (zipCode == null || zipCode.length() != 5) {
        throw new IllegalArgumentException("Wrong zip code");
      }
    }
  }
}
```

Übliche Architekturen

- Validatoren für jedes
  - Framework
  - Schicht
  - System
- Validierung als Teil der Anwendungslogik

holisticon
MEMBER OF SIGMA GROUP

Bean Validation JSR-303

Bean Validation

- Idee
  - Validierungsregeln an Daten
  - Offener JCP Standard (JSR-303), Teil von JEE6
  - Open Source Implementierung
  - Definiert eine Menge von Constraints
  - Definiert eine Validation API

```
public class ZipCode {
  @Pattern(regexp = "[0-9]*")
  @Size(min=5, max=5)
  private String content;
}
```

- Standard Constraint Annotationen

```
@AssertFalse
@AssertTrue
@DecimalMin(value)
@DecimalMax(calue)
@Digits(integer, fraction)
@Future
@Min(value)
@Max(value)
@NotNull
@Null
@Past
@Pattern(regexp, flags)
@Size(min, max)
@Valid
```

Bean Validation

Bean Validation

Validation API

```
public class ValidatorHelper {

  private static Validator validator =
   Validation.buildDefaultValidatorFactory().getValidator();

  public static void validate(Object object) {
    Set<ConstraintViolation<Object>> violations = validator.validate(object);
    if (violations.size() > 0) {
      StringBuffer buf = new StringBuffer("constraint violations in")
       .append(object.getClass().getName() + "[");
      for (ConstraintViolation<Object> violation : violations) {
        buf.append("\"").append(violation.getPropertyPath()).append("\" ")
          .append(violation.getMessage()).append(" (invalidValue=\"")
          .append(violation.getInvalidValue()).append("\"); ");
      }
      throw new IllegalArgumentException(buf.append("]").toString());
    }
  }
}
```

Bean Validation

Eigene Constraints

```
@Target({ FIELD, PARAMETER })
@Retention(RUNTIME)
@Constraint(validatedBy = {})
@Pattern(regexp = "[0-9]*")
@Size(min = 5, max = 5)
@ReportAsSingleViolation
public @interface ZipcodeConstraint {

  String message() default "{Zipcode.invalid_zipcode}";

  Class<?>[] groups() default {};

  Class<? extends Payload>[] payload() default {};
}
```

Bean Validation

Eigener Validator

```
@Constraint(validatedBy = {ZipcodeValidator.class})
public @interface ZipcodeConstraint {
  // …
}

public class ZipcodeValidator implements
 ConstraintValidator<ZipcodeConstraint, String> {
  private String message;
  public void initialize(ZipcodeConstraint annotation) {
    message = annotation.message();
  }
  public boolean isValid(String value, ConstraintValidatorContext ctx) {
    if (value.length() != 5) {
      ctx.buildConstraintViolationWithTemplate(message)
       .addConstraintViolation();
      return false;
    }
    return true;
  }
}
```

Bean Validation

Validation Groups, Constraint List

```
public class ZipCode {
  @Pattern.List({
    @Pattern(regexp = "[0-9]*"),
    @Pattern(regexp = "[0-9\\*]*", groups = { Search.class })
  })
  @Size.List({
    @Size(min=5, max=5),
    @Size(min=1, max=5, groups = { Search.class })
  })
  @NotNull(groups = { Default.class, Search.class })
  private String content;
}

public class ValidatorHelper {
  public static void validate(Class<?>[] groups, Object object) {
  Set<ConstraintViolation<Object>> violations =
   validator.validate(object, groups);
   // …
  }
}
```

Bean Validation

- JSF 2
  - Zero-config Standardvalidator
- JPA
  - Mittels JPA Hooks
  - Spezielle Graph-Traverser

holisticon
MEMBER OF SIGMA GROUP

Domänenspezifische Datentypen

Domänenspezifische Typen


```
/**
 * Domain specific type
 */
public class Customer {


  @Valid
  @NotNull
  private Name name;

  @Enumerated(EnumType.STRING)
  @NotNull
  private Gender gender;

  // …
}
```

Domänenspezifische Typen


```
/**
 * Domain specific type
 */
public class Customer {

  // …
  @Valid
  @NotNull
  private Address postalAddress;


  @Valid
  @NotNull
  private Address billingAddress;

  // …
}
```

Domänenspezifische Typen


```
public class Address {

  private String addressLine1;

  private String addressLine2;

  @Valid
  private ZipCode zip;

  @Valid
  private City city;

}
```

Domänenspezifische Typen

Dipl.-Ing. (FH)
Oliver Ochs
Senior Consultant

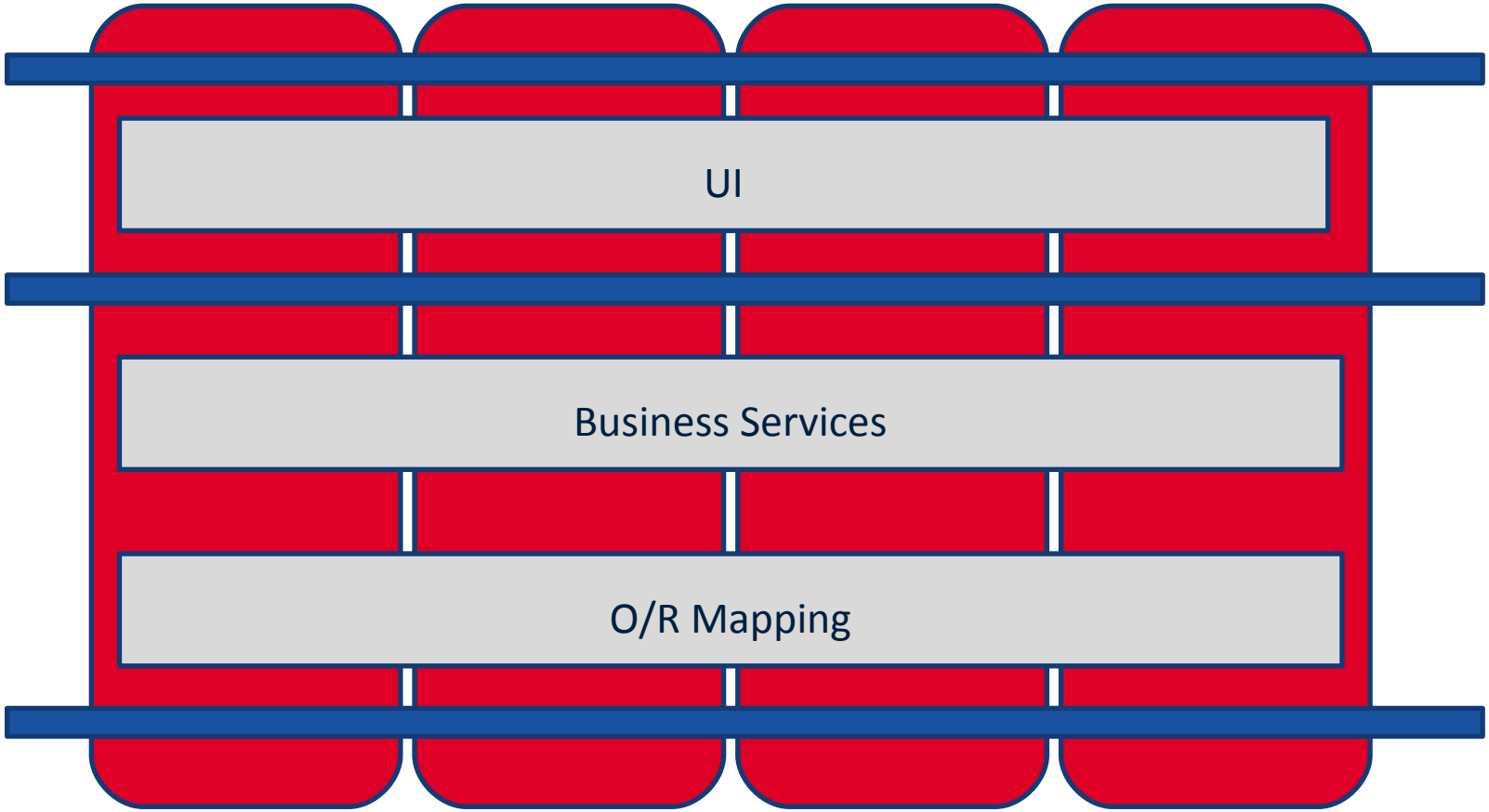
holisticon
Holisticon AG
Friedensallee 120
22763 Hamburg
www.holisticon.de
Mobile: +49 171 975 0176
Office: +49 40 5074 2722
Fax: +49 40 5074 2724
E-Mail: oliver.ochs@holisticon.de

```
public class ZipCode {

  @Pattern.List({
    @Pattern(regexp = "[0-9]*"),
    @Pattern(regexp = "[0-9\\*]*",
     groups = Search.class)
  })
  @Size(min=1, max=5)
  @NotNull(groups = {
    Default.class, Search.class})
  @Column(name = "zip")
  private String content;

  // …
}
```

Domänenspezifische Typen

- Domänenspezifische Datentypen
  - Datencontainer mit Semantik
  - Stellt schichtenübergreifende Typisierung sicher
- Konvertierung
  - Konvertierung für UI (String <-> Object <-> String)
  - Konvertierung für DB (O/R-Mapping)
- Validierung
  - Ist eng mit Format / Semantik der Daten verbunden
  - Muss in die Frameworks integriert werden

Domänenspezifische Typen



Domänenspezifische Typen

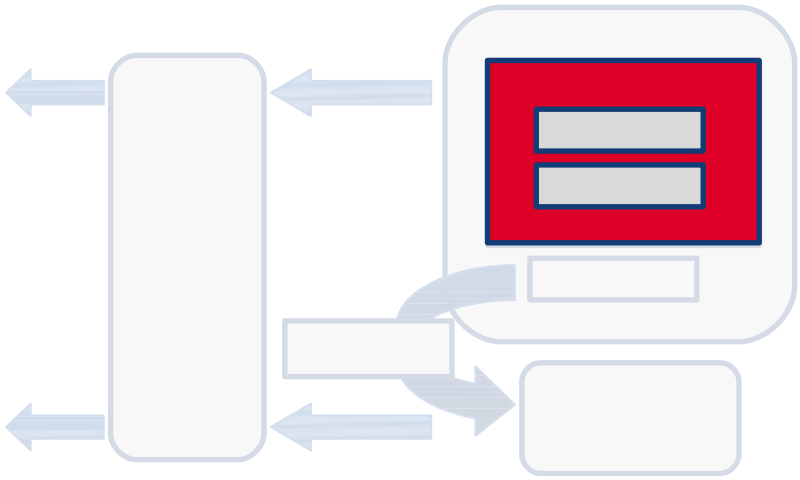
- Domänenspezifische Typen mit JSR-303?
  - Konvertierung
  - Validierung

holisticon
MEMBER OF SIGMA GROUP

Spring MVC Integration

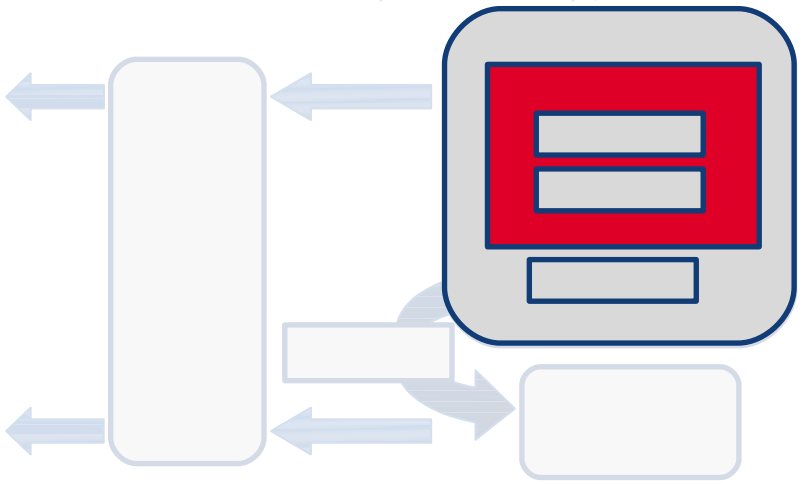
Property Editor Support (Konverter)

```
public class ZipCodeBinder extends PropertyEditorSupport {

  public String getAsText() {
    return (getValue() == null ? "" : ((ZipCode) getValue()).getContent());
  }

  public void setAsText(final String text) throws IllegalArgumentException {
    if (text != null && !text.isEmpty()) {
      ZipCode zipCode = new ZipCode(text);
      setValue(zipCode);
    } else {
      setValue(null);
    }
  }
}
```



Controller

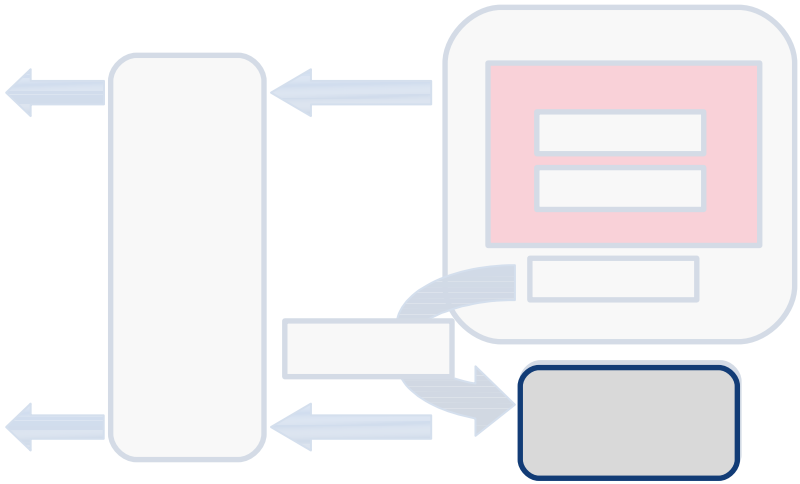
```
@RequestMapping(value="/customer.html", method = RequestMethod.PUT)
public String processForm(@Valid Customer customer, BindingResult result,
 Map<String, Object> model) {

  if (result.hasErrors()) {
    // […]
    return "form";
  }
  Customer savedCustomer = customerService.createCustomer(customer);
  // […]
  model.put("customer", savedCustomer);
  return "form";
}
```



Spring MVC

View

```
<form:form method="put" action="customer.html" commandName="customer">

  <fieldset>
    <form:label path="postalAddress.zip„>PLZ:</form:label>
    <form:input path="postalAddress.zip" />
    <form:errors path="postalAddress.zip" />
    <input type="submit" value="Create" />
  </fieldset>

</form:form>
```



holisticon
MEMBER OF SIGMA GROUP

JSF2 Integration

JSF Integration

- Apache MyFaces Extensions Validator (ExtVal)
- Konverter

```
@FacesConverter(forClass=ZipCode.class)
public class ZipcodeConverter implements Converter {

  public Object getAsObject(FacesContext ctx, UIComponent comp, String value) {
    if (value == null)
      return null;
    return new ZipCode(value);
  }

  public String getAsString(FacesContext ctx, UIComponent comp, Object value) {
    return ((ZipCode)value).getContent();
  }
}
```

JSF Integration

- Demo

holisticon
MEMBER OF SIGMA GROUP

JFace Data Binding Integration

JFace Data Binding

Binding

```
public class ZipCodeComposite extends Composite {

  public void createBinding(DataBindingContext dbc) {

    // […]

    UpdateValueStrategy t2m = new UpdateValueStrategy();
    UpdateValueStrategy m2t = new UpdateValueStrategy();
    m2t.setConverter(new ZipCode2StringConverter());
    t2m.setConverter(new String2ZipCodeConverter());

    t2m.setAfterConvertValidator(new BeanValidator());

    dbc.bindValue(observableTarget, observableModel, t2m, m2t);
  }
}
```

JFace Data Binding

IConverter

```
public class ZipCode2StringConverter implements IConverter {

  public Object getToType() {
    return String.class;
  }

  public Object getFromType() {
    return ZipCode.class;
  }

  public Object convert(Object fromObject) {
    ZipCode zipCode = (ZipCode) fromObject;
    if (zipCode == null) {
      return null;
    }
    return zipCode.getContent();
  }
}
```

JFace Data Binding

Validator

```
public class BeanValidator implements IValidator {

  private ValidatorFactory factory = Validation.buildDefaultValidatorFactory();

  public IStatus validate(Object value) {
    Set<ConstraintViolation<Object>> violations = factory.getValidator()
     .validate(value, new Class<?>[] { Default.class });

    if (violations.size() > 0) {
      List<IStatus> statusList = new ArrayList<IStatus>();
      for (ConstraintViolation<Object> cv : violations) {
        statusList.add(ValidationStatus.error(cv.getMessage()));
      }
      return new MultiStatus(Activator.PLUGIN_ID, IStatus.ERROR, statusList
       .toArray(new IStatus[statusList.size()]), "Validation errors", null);
    }

    return ValidationStatus.ok();
  }
}
```

Domänenspezifische Typen

- Erkenntnisse
  - Wiederverwendbarkeit
  - Semantische Typsicherheit
  - Validierungslogikkapselung
  - Trennung der Verantwortlichkeit

  - Spezielle Konverter
  - Konverter verletzen die Kapselungsgrenzen

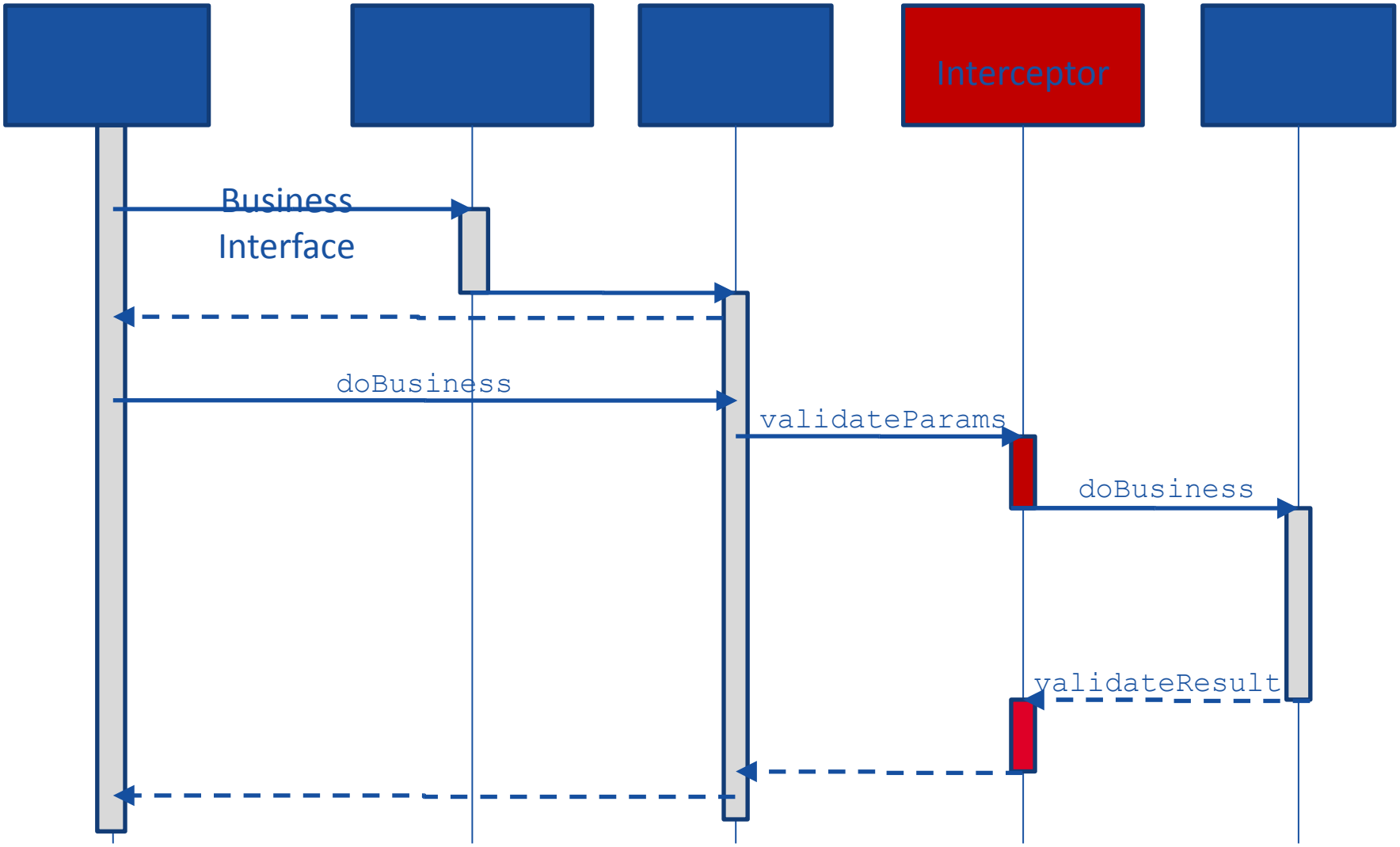
holisticon
MEMBER OF SIGMA GROUP

Service Boundary Validation

Service Boundary Validation

- Motivation
  - Richtlinien für Schnittstellendesign
  - Design-by-Contract
  - Transparenz für den Entwickler
- Grundprinzip
  - Aspekt / Interceptor
  - Validierung der Methodenparameter
  - Validierung der Rückgabewerte

Service Boundary Validation



Service Boundary Validation

EJB Interceptor

```
public class ServiceValidationInterceptor {

  @AroundInvoke
  public Object validate(InvocationContext context) throws Exception {

    Method method = context.getMethod();
    Object[] args = context.getParameters();

    validateParams(args, method);

    Object result = context.proceed();

    if (result != null) {
      validateResult(result);
    }

    return result;
  }
}
```

Service Boundary Validation

Spring Aspekt

```
public class ServiceValidationAspect {
  public void validateArguments(JoinPoint joinPoint) {
    Object[] args = joinPoint.getArgs();
    Method method = ((MethodSignature) joinPoint.getSignature()).getMethod();
    …
  }
  public void validateResult(JoinPoint joinPoint, Object retVal) { … }
}
```

```
<beans[…] >
  <bean id="bvAspect" class="….aspect.ServiceValidationAspect" />
  <aop:config>
    <aop:pointcut id="bv" expression="execution(* *.*(..))" />
    <aop:aspect ref="bvAspect">
      <aop:before pointcut-ref="bv" method="validateArguments" />
      <aop:after-returning pointcut-ref="bv" returning="retVal"
       method="validateResult" />
    </aop:aspect>
  </aop:config>
</beans>
```

Service Boundary Validation

Was ist mit Gruppenvalidierung?

```
public interface Search extends ValidationGroup {

}

@Target({ ElementType.METHOD })
@Retention(RetentionPolicy.RUNTIME)
public @interface ValidationAdvice {

  Class<? extends ValidationGroup>[] value();

}

public interface CustomerService {

  public List<Customer> getAllCustomers();

  @ValidationAdvice(Search.class)
  public List<Customer> searchCustomer(CustomerSearchParameters criteria);
}
```

Service Boundary Validation

Gruppenvalidierung

```
public void validateArguments(JoinPoint joinPoint) {

  Object[] args = joinPoint.getArgs();
  Method method = ((MethodSignature) joinPoint.getSignature()).getMethod();

  Class<?> groups = new Class<?>[] { Default.class };

  ValidationAdvice advice = method.getAnnotation(ValidationAdvice.class);
  if (advice != null) {
    groups = advice.value();
  }

  ValidatorHelper.requireValidated(groups, args);
}
```

holisticon
MEMBER OF SIGMA GROUP

Offene Punkte

Offene Punkte und Ausblick

- JSR-303 Bean Validation ist praxistauglich!
- Sinnvolle Ergänzung: Apache MyFaces ExtVal!
- Domänenspezifische Typen zur Kapselung!

- Was ist mit Konvertern?
- Conditional/Cross Validation mit EL?
- Was ist mit JAX-WS/JAX-B?

Vielen Dank!

http://blog.holisticon.de/