NOTES & DOMINO
ENTWICKLERCAMP

assono

Pages und Java

EntwicklerCamp
12. März 2013

Innovative Software-Lösungen.

www.assono.de

NOTES & DOMINO ENTWICKLERCAMP

assono

Bernd Hort

Diplom-Informatiker, Universität Hamburg

seit 1995 entwickle ich Lotus Notes Anwendungen

IBM Certified Application Developer, System Administrator & Instructor

Sprecher auf diversen Konferenzen & Lotusphere 2008

bhort@assono.de
http://www.assono.de/blog
040/73 44 28-315

assono
IT-Consulting & Solutions

Agenda

- Motivation
- Eigene Java-Klassen
- JavaServer Faces Lifecycle
- Managed Beans
- Nützliche JSF-Elemente / Begrifflichkeiten
- Model-View-Controller Pattern
- Fragen & Antworten

Motivation

- Besseres Verständnis für XPages
- Debugger
- Objekt-Orientierung
- Model-View-Controller
- Komplexität
- Performance
- Open-Source-Projekts
- Java macht sich gut im Lebenslauf ;-)

NOTES & DOMINO ENTWICKLERCAMP

assono

Java in Server Side JavaScript (SSJS)

- Java Klassen können direkt im SSJS verwendet werden
- Aufruf mit vollem Java-Package-Pfad

```
var now : java.util.Date = new java.util.Date();
```

- Alternativ mit importPackage

```
importPackage(de.assono.ec13);
var myTest = new Test();
return myTest.dummy();
```

NOTES & DOMINO ENTWICKLERCAMP

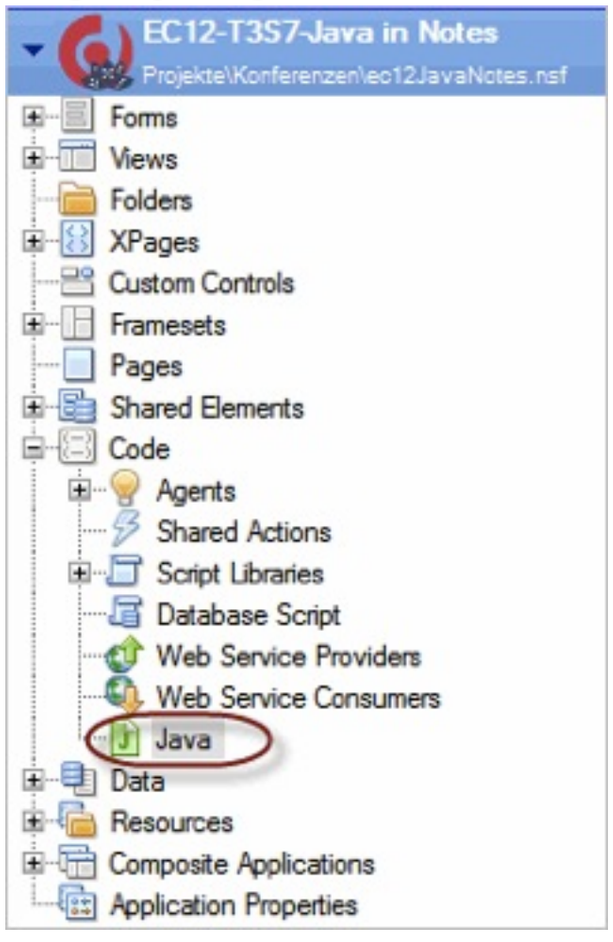
assono

Hello World-XPage

```
<xp:panel xp:key="mainContent">
   <xp:text escape="true"   id="computedField2" tagName="h1">
       <xp:this.value><![CDATA[#{javascript:var
myFirstClass:de.assono.ec13.MyFirstClass = new
de.assono.ec13.MyFirstClass();
myFirstClass.getCreetings();}]]></xp:this.value>
    </xp:text>
    <xp:text escape="false" id="computedField1">
        <xp:this.value><![CDATA[#{javascript:
           var result : String = "Aktueller Benutzer: " +
@Name('[CN]',@UserName()) + "<br />";
           result += "Aktuelle Datenbank: " + database.getTitle() +
"<br />";
           var now : java.util.Date = new java.util.Date();
           var formatter : java.text.SimpleDateFormat =
              new java.text.SimpleDateFormat("'Heute ist 'EEEE',
der 'dd.MM.yyyy'. Es ist jetzt 'HH:mm' Uhr.'", Locale.GERMANY);
           result += "Jetzt ist es: " + formatter.format(now) +
"<br />";
           return result}]]></xp:this.value>
              </xp:text>
           </xp:panel>
```

Eigene Java-Klassen in XPages

- Vor Lotus Domino 8.5.3
  - In der Java-Perspektive im Ordner WebContent\WEB-INF\src ablegen und Ordner im Build Path definieren (Beispiel folgt)
- Lotus Domino 8.5.3 (und folgende)
  - Neues Design Element

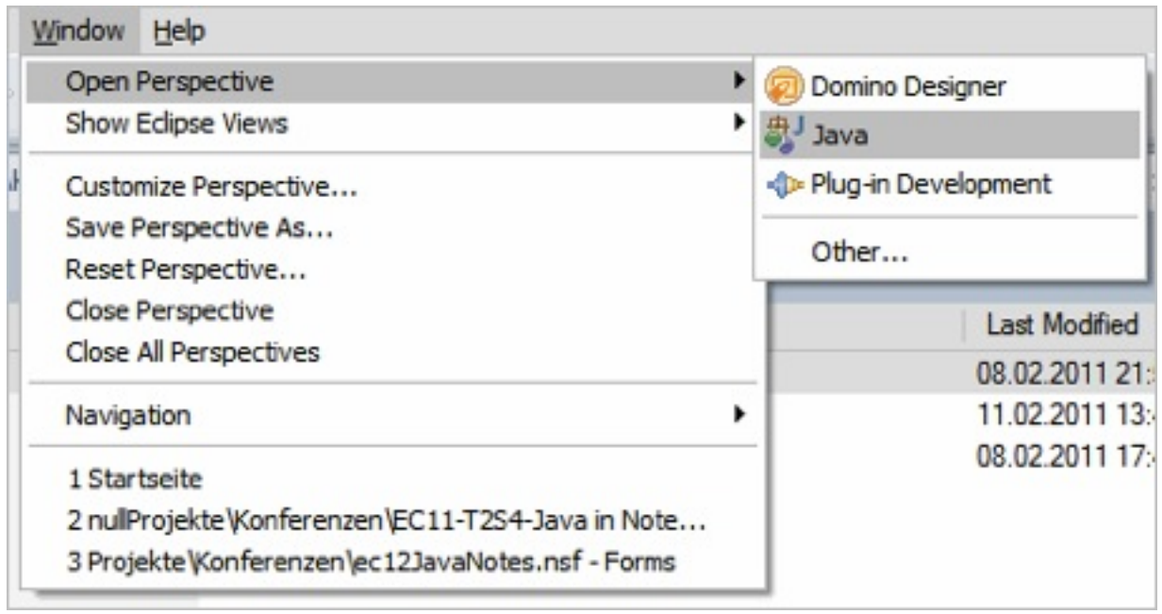


Java-Klassen für XPages nicht sichtbar für Agenten

- Für XPages geschriebene Java-Klassen sind nicht für Java-Agents sichtbar.
- Die Ursache sind unterschiedliche ClassLoader-Mechanismen.
- Daran wird sich auf absehbaren Zeit nichts ändern!

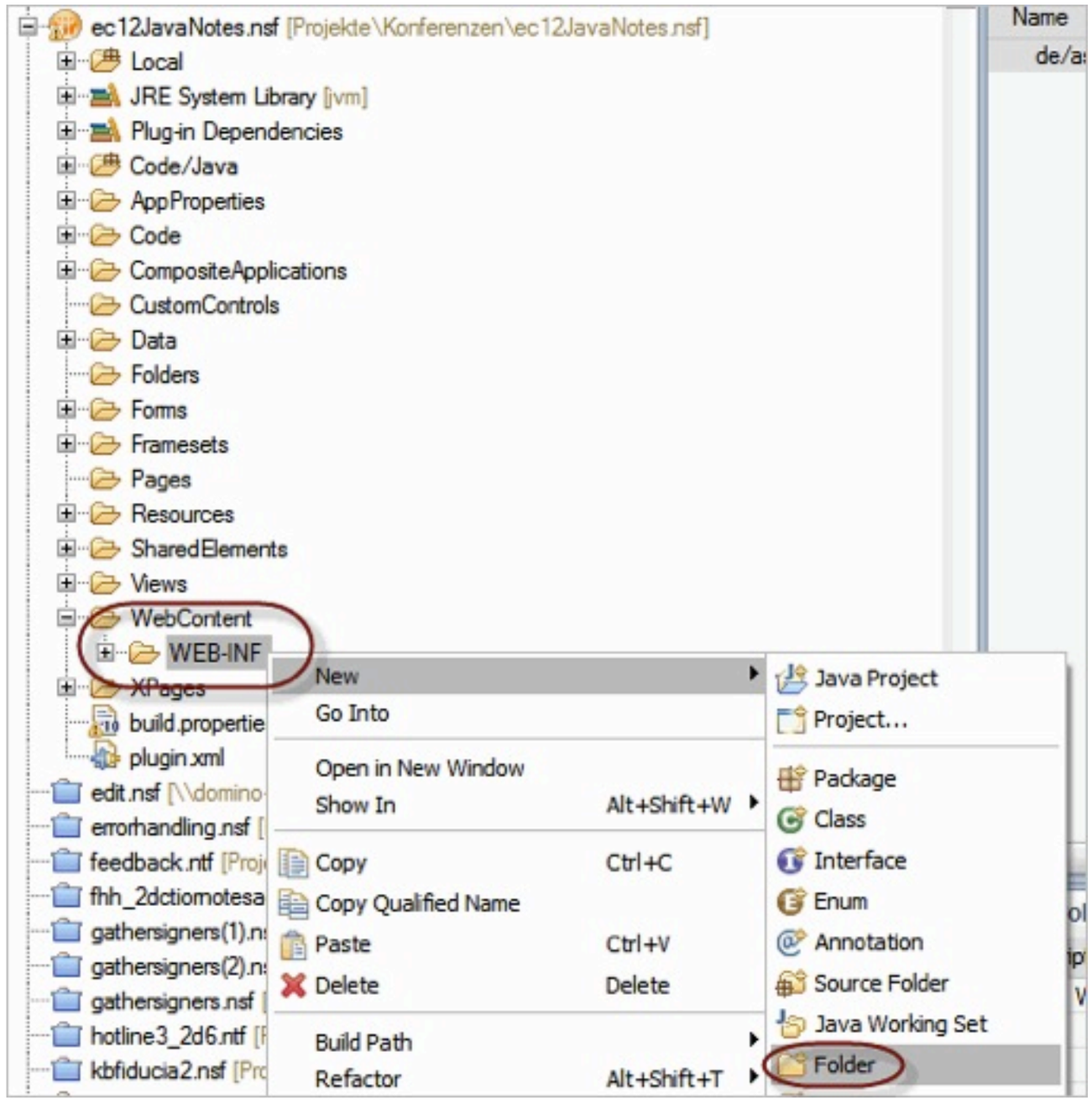
Eigene Java-Klassen in <= Domino 8.5.2

- In die Java-Perspektive wechseln



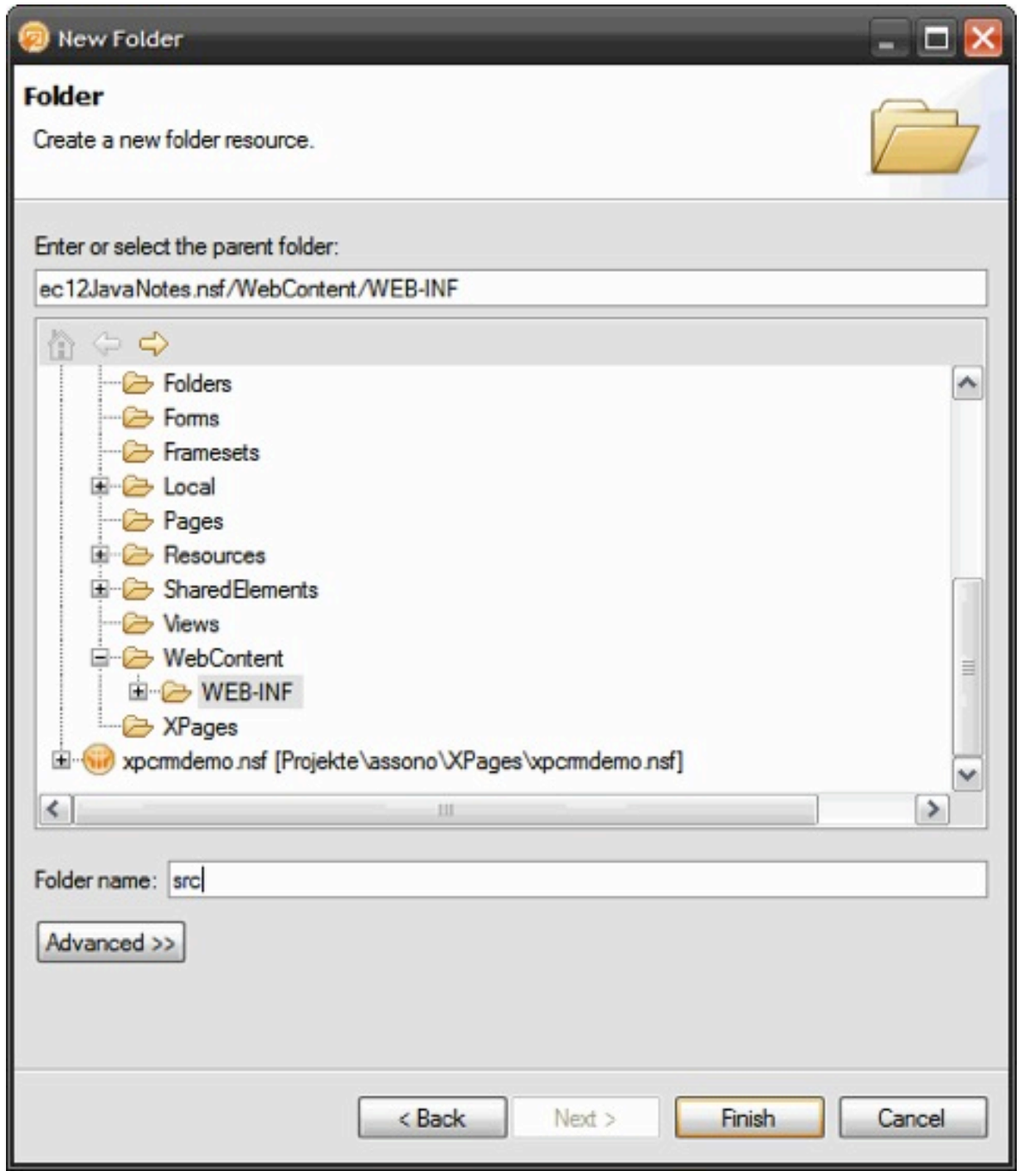
Eigene Java-Klassen in <= Domino 8.5.2 (forts.)

- Unter WebContent\WEB-INF New\Folder auswählen



Eigene Java-Klassen in <= Domino 8.5.2 (forts.)

- Folder name „src“

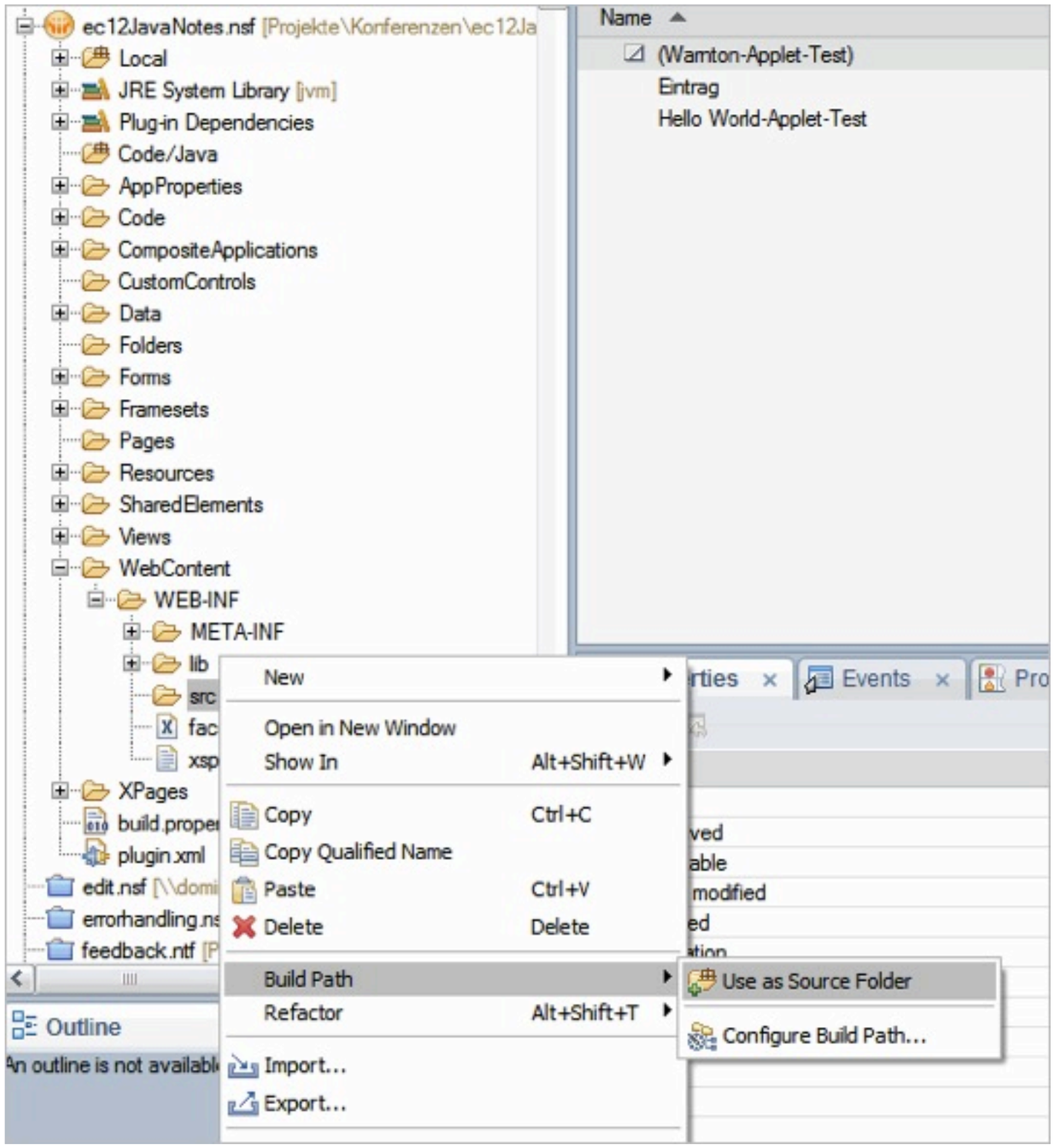


NOTES & DOMINO ENTWICKLERCAMP

assono

Eigene Java-Klassen in <= Domino 8.5.2 (forts.)

- Neuen Ordner auswählen
- Im Kontext-Menü Build Path „Use as Source Folder“ auswählen

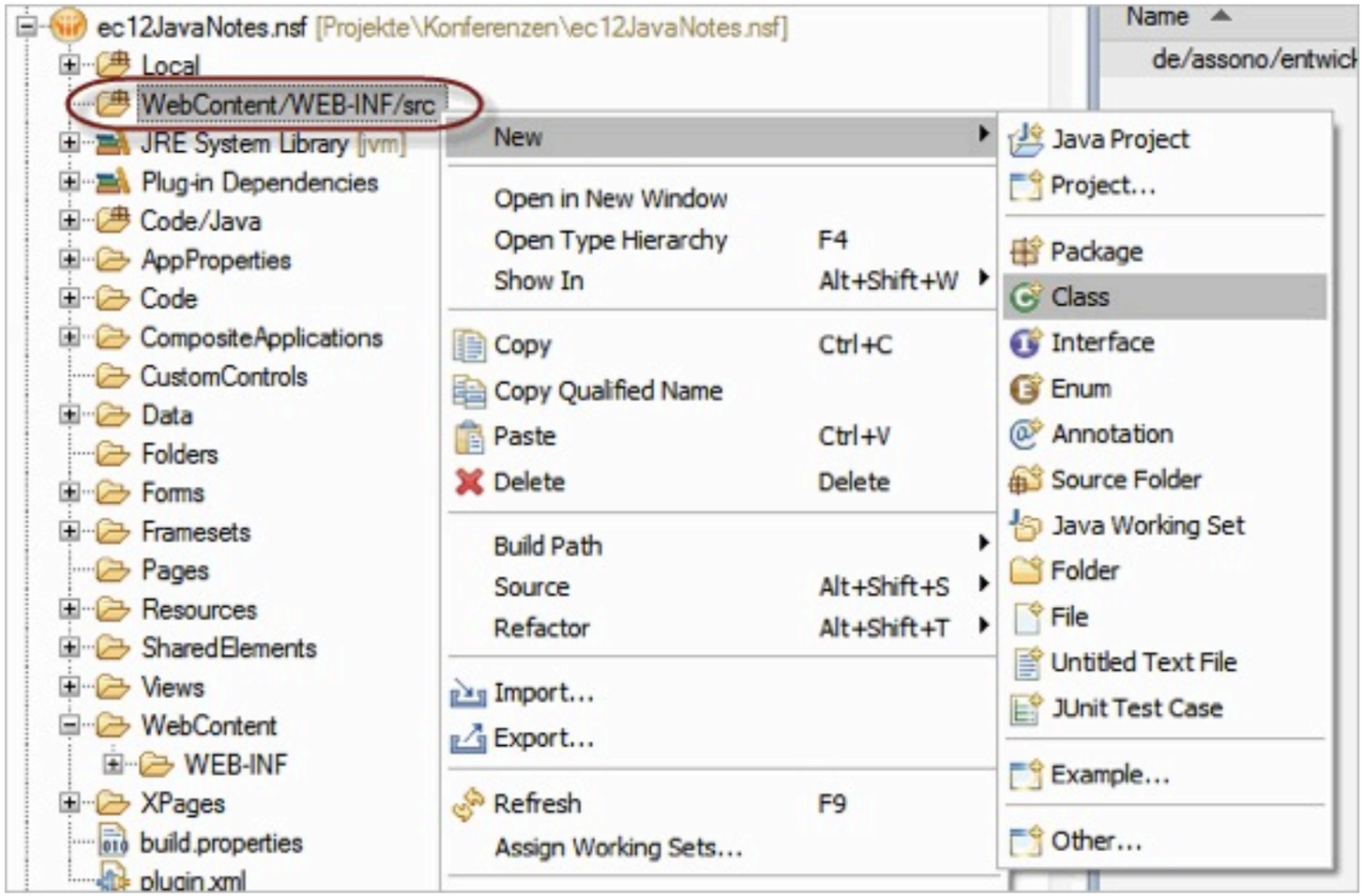


Eigene Java-Klassen in <= Domino 8.5.2 (forts.)

- Der Ordner wird automatisch nach oben verschoben

Eigene Java-Klassen in <= Domino 8.5.2 (forts.)

- Den Ordner markieren
- Im Kontextmenü New\Class auswählen

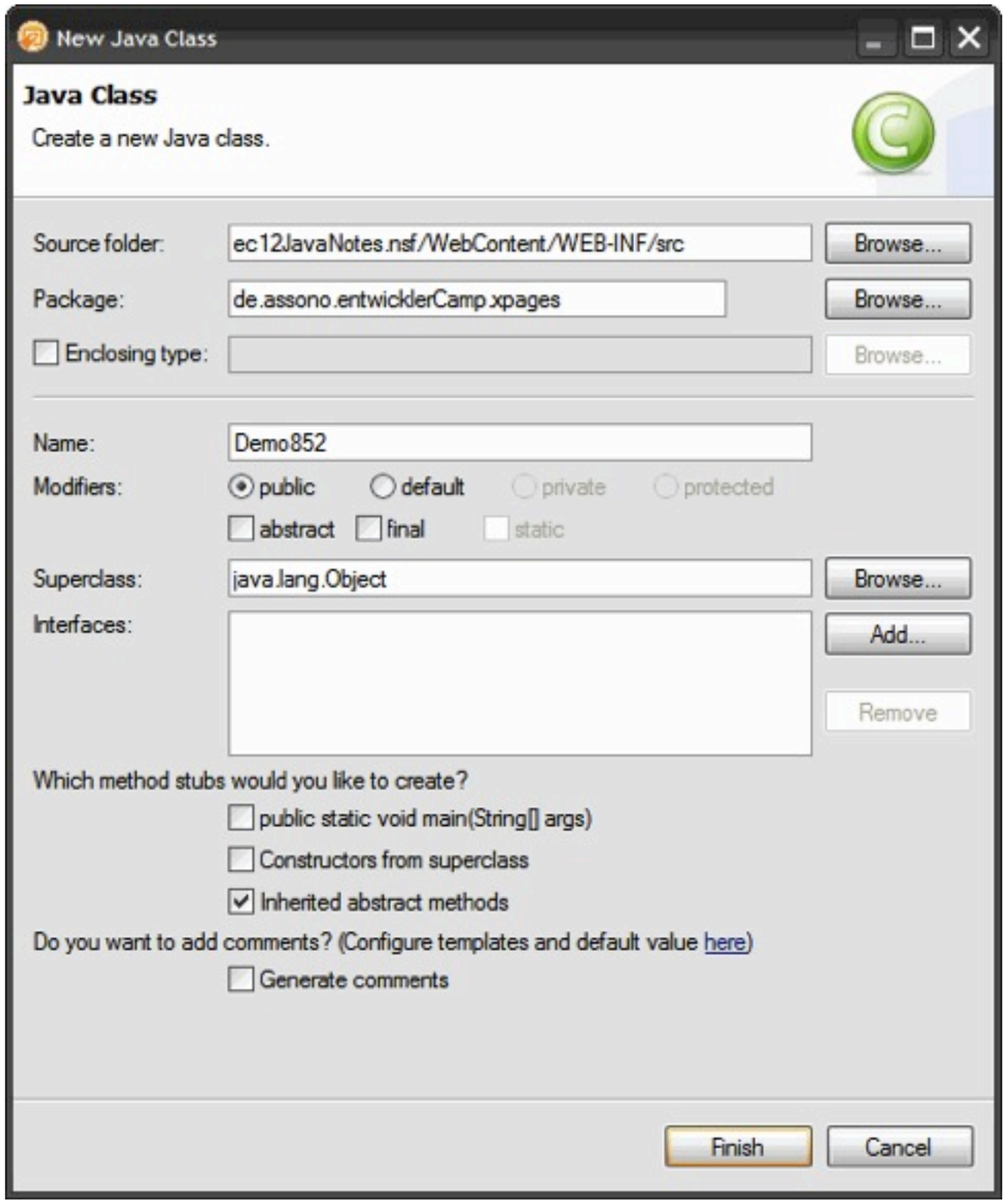


NOTES & DOMINO ENTWICKLERCAMP

assono

Eigene Java-Klassen in <= Domino 8.5.2 (forts.)

- Package-Namen angeben
- Klassen-Namen angeben
- Finish



«XPage»
HelloWorld

Recycling

- **Recycling ist erste (Java-)Bürgerpflicht!**
- Für jedes Java-Objekt (Session, Database, View, Document usw.) gibt es im Hintergrund ein C-Objekt, dass ohne recycle() nie mehr frei gegeben wird.
- Das Nummer 1-Problem für Neulinge (und manchmal auch für „alte Hasen“.).
- Abhängige Objekte werden automatisch recycled, wenn die Eltern-Objekte recycled werden. (ViewEntry =>View)
- Achtung! DateTime ist Kind-Objekt von Session.
  Separat recyclen!

Notes.ini Variable HTTPJVMMaxHeapSize

- Mit Version 8.5.1 wurde eine neue Notes.ini Variable für den Domino Server eingeführt
  HTTPJVMMaxHeapSize
  um den maximal genutzten Speicher zu definieren

- Mit 8.5.2 wurde diese Variable per default auf
  HTTPJVMMaxHeapSize=64M
  gesetzt

- Das ist entschieden zu wenig für XPages!

- Gilt nur für 32-bit Plattformen

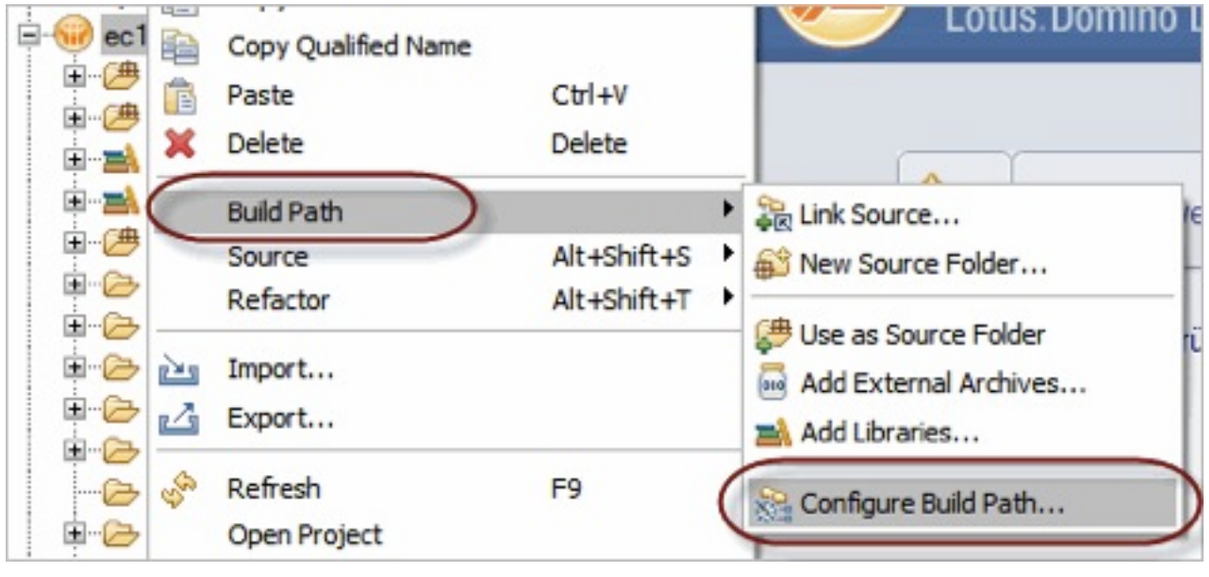
- IBM Lotus Support Dokument
  http://www-01.ibm.com/support/docview.wss?uid=swg21377202

Jar-Dateien in XPages

- Im Ordner WebContent/WEB-INF/lib ablegen
  - Jar-Datei per Drag & Drop im Ordner ablegen
  - Über den Build-Path einbinden (Beispiel folgt)
- Ab 9.0 neues Design-Element jar
- Alternativ im Designer und Server unter {Programmverzeichnis}\jvm\lib\ext ablegen
- Achtung! Wird die jar-Datei ausgetauscht, muss der HTTP-Task neu gestartet werden!
  - Restart task http

Build-Path konfigurieren

- Im Designer in den Eclipse View „Packet Explorer“ wechseln
- Rechte Maustaste auf Build Path

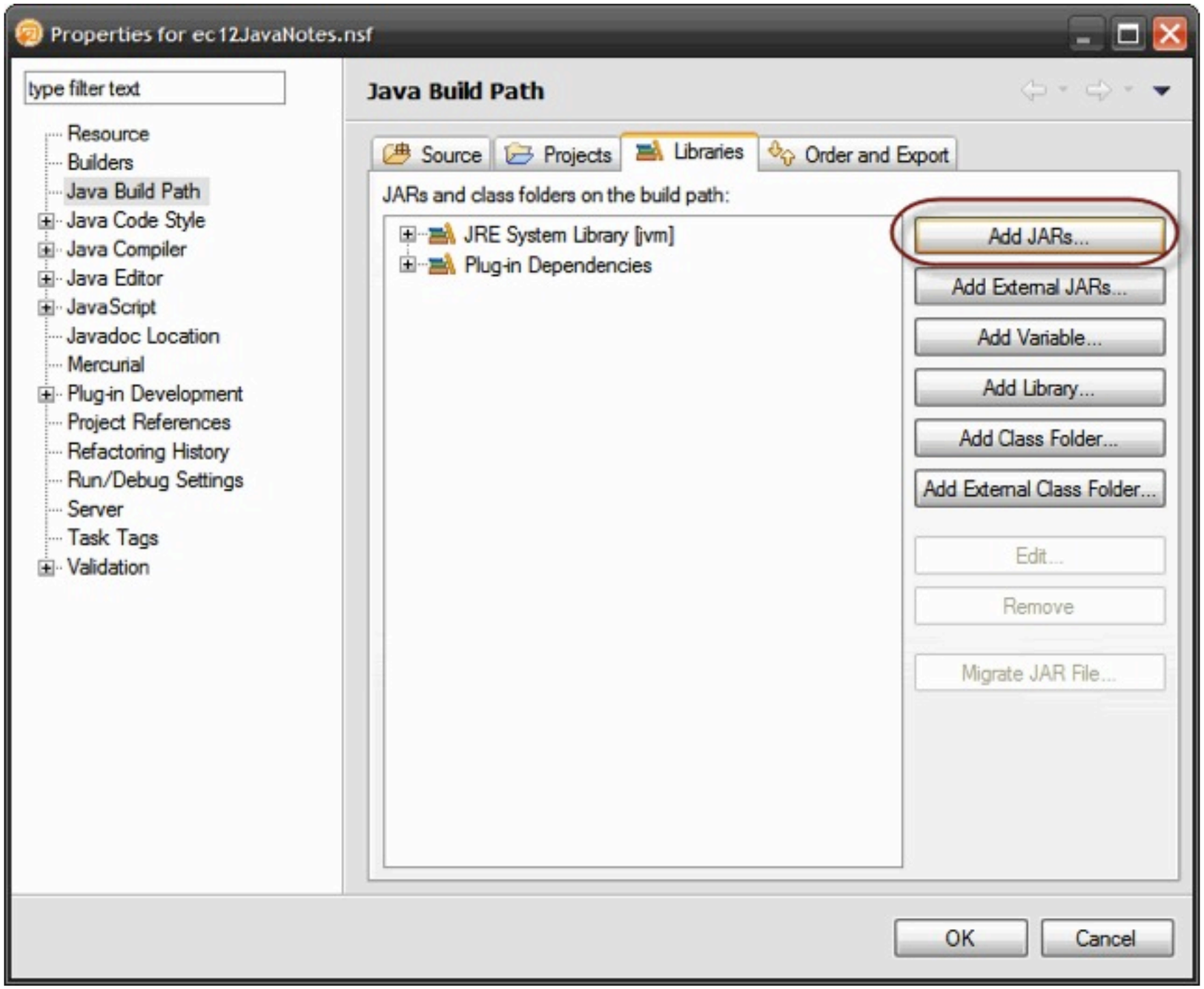


NOTES & DOMINO ENTWICKLERCAMP

assono

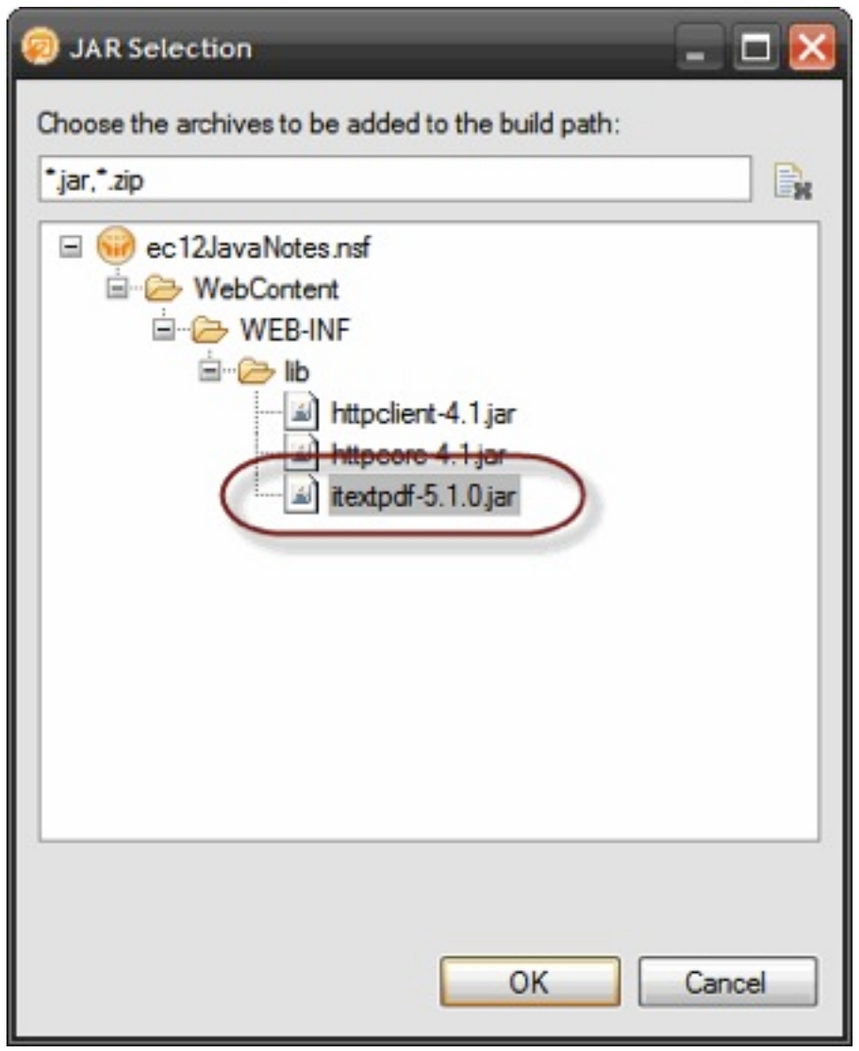
Build-Path konfigurieren

- Libraries
- Add JARs...



Build-Path konfigurieren

- Jar-Datei auswählen ;-)

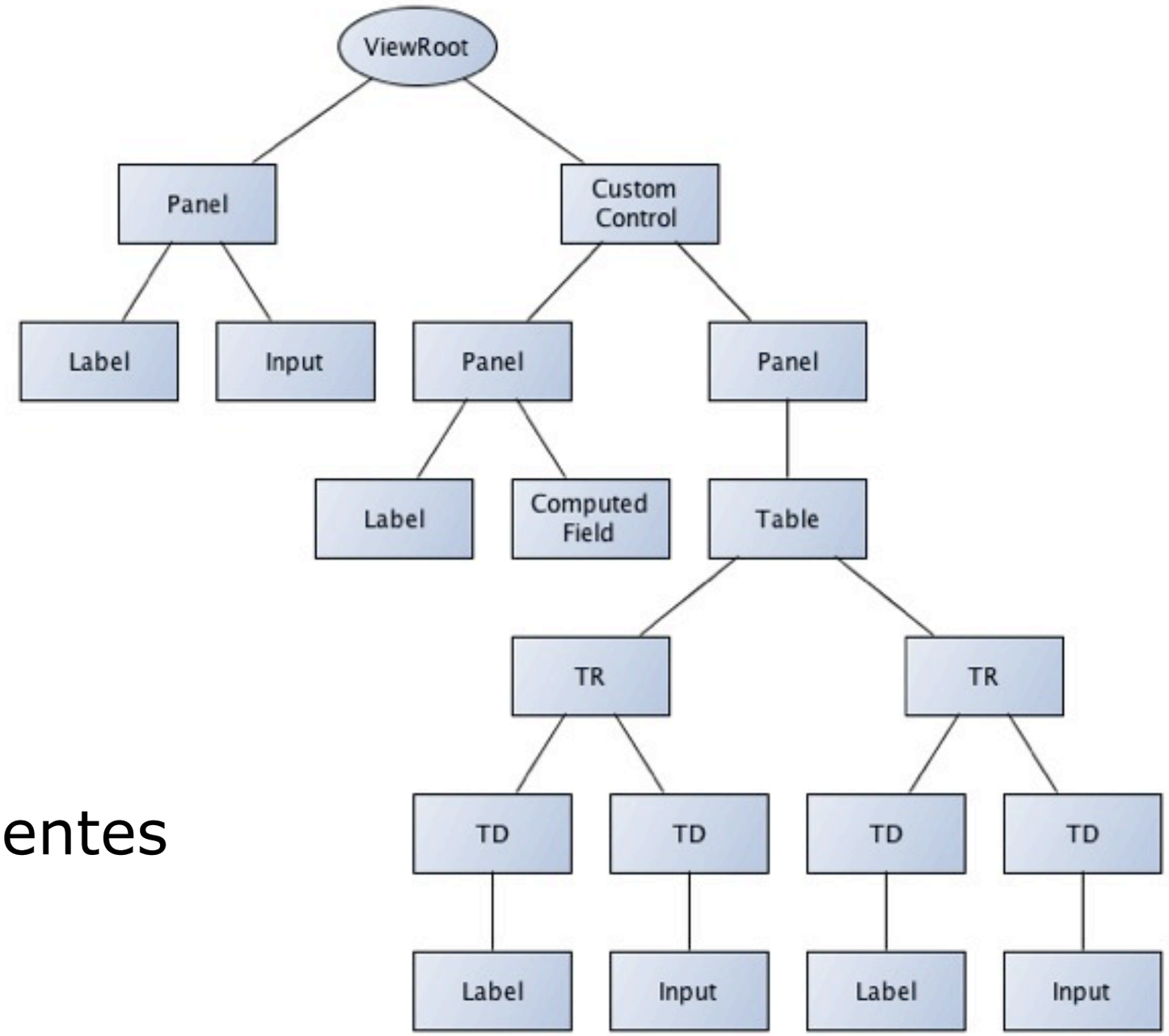


JavaServer Faces

- XPages basieren auf JavaServer Faces (JSF)
- Basis ist JSF 1.1
  - Ziemlich alt / aktuell ist 3.0
  - Aber viele Erweiterung der aktuellen Version integriert
- Aus der XML-Definition einer XPage wird eine Java-Klasse generiert.
- Die Java-Klasse wird zur Laufzeit ausgeführt.

XPage intern als Objektbaum

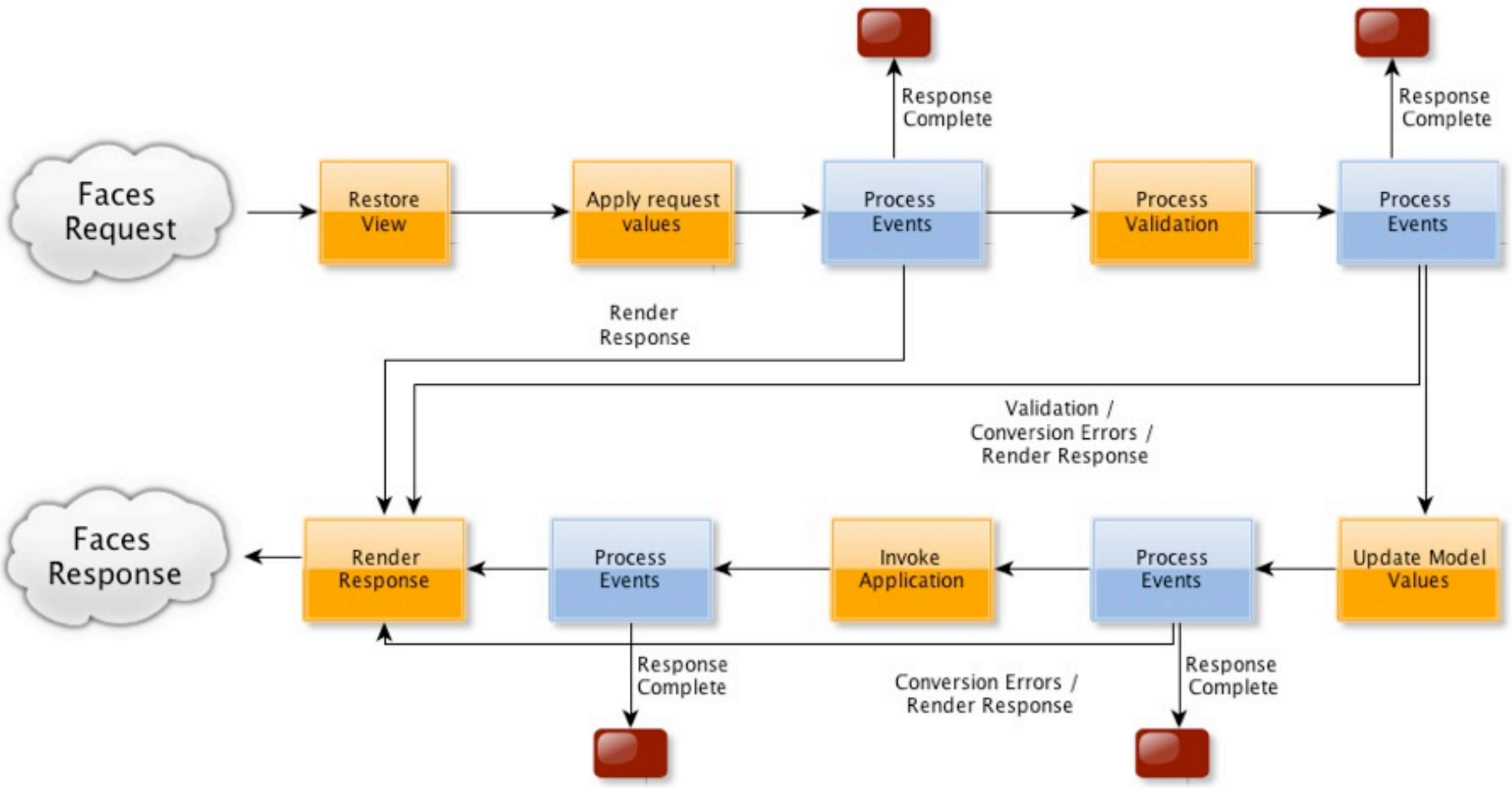
- ViewRoot als Wurzel
- Jedes Element auf der Webpage ist ein Objekt
- Verschachtelung wird hierarchisch abgebildet
- Zustand jedes Elementes wird gespeichert



NOTES & DOMINO ENTWICKLERCAMP

assono

JSF Lifecycle



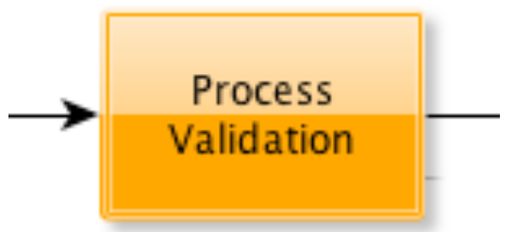
JSF Lifecycle – Restore View

- Der Objektbaum wird (wieder) hergestellt.
- Der vorherige Status aller Elemente wird eingetragen.

JSF Lifecycle – Apply request values

- Die vom Browser übermittelten Strings werden den Eingabeelementen zugewiesen.
- Konvertierung der Strings in die Datentypen, die den Eingabeelementen zugewiesenen worden sind
  - Gilt nur für einfache Datentypen
  - Bei Notes Items findet die Konvertierung erst in der nächsten Phase statt

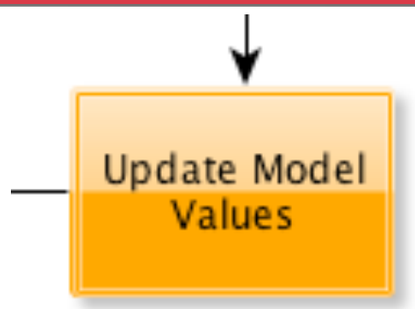
JSF Lifecycle – Process Validation



- Überprüfung der Eingabe auf Gültigkeit

- Zunächst werden eventuell vorhandene Converter angewendet
- Überprüfung aller Validatoren
- Zu diesem Zeitpunkt sind die Werte noch nicht im DominoDocument.
  - document1.getItemValue("") liefert noch den alten Wert zurück
  - inputElement.getSubmittedValue() gibt die übermittelten Werte zurück

JSF Lifecycle – Update Model Values



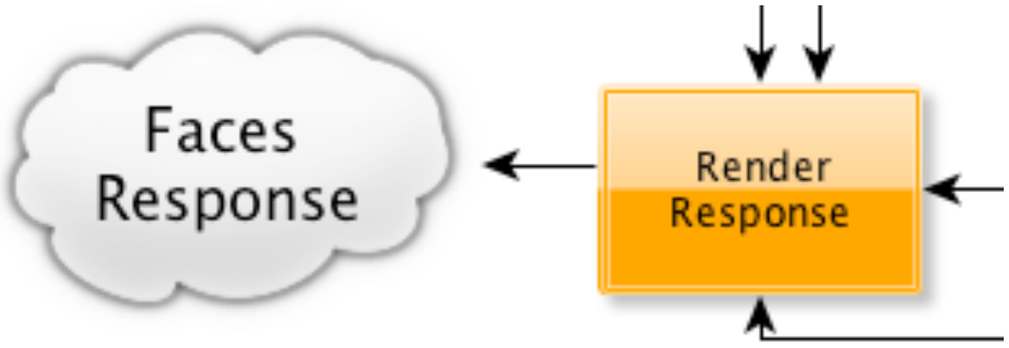
- In dieser Phase werden die Werte den Items im DominoDocument zugewiesen
  - Aufruf der Methode setValue für jedes Eingabeelement
  - SubmittedValue wird auf null gesetzt

JSF Lifecycle – Invoke Application

- Ausführen der mit dem Event verbundenen Aktionen

```
<xp:eventHandler event="onclick" submit="true" refreshMode="complete">
    <xp:this.action>
        <xp:actionGroup>
            <xp:confirm
                message="Wollen Sie wirklich?"></xp:confirm>
            <xp:executeScript>
                <xp:this.script><![CDATA[#{javascript:var demo:de.assono.ec13.Demo =
                new de.assono.ec13.Demo();
                demo.doFancyStuff();}]]></xp:this.script>
            </xp:executeScript>
            <xp:openPage
                name="/HelloWorld.xsp"></xp:openPage>
    </xp:actionGroup>
    </xp:this.action>
</xp:eventHandler>
```

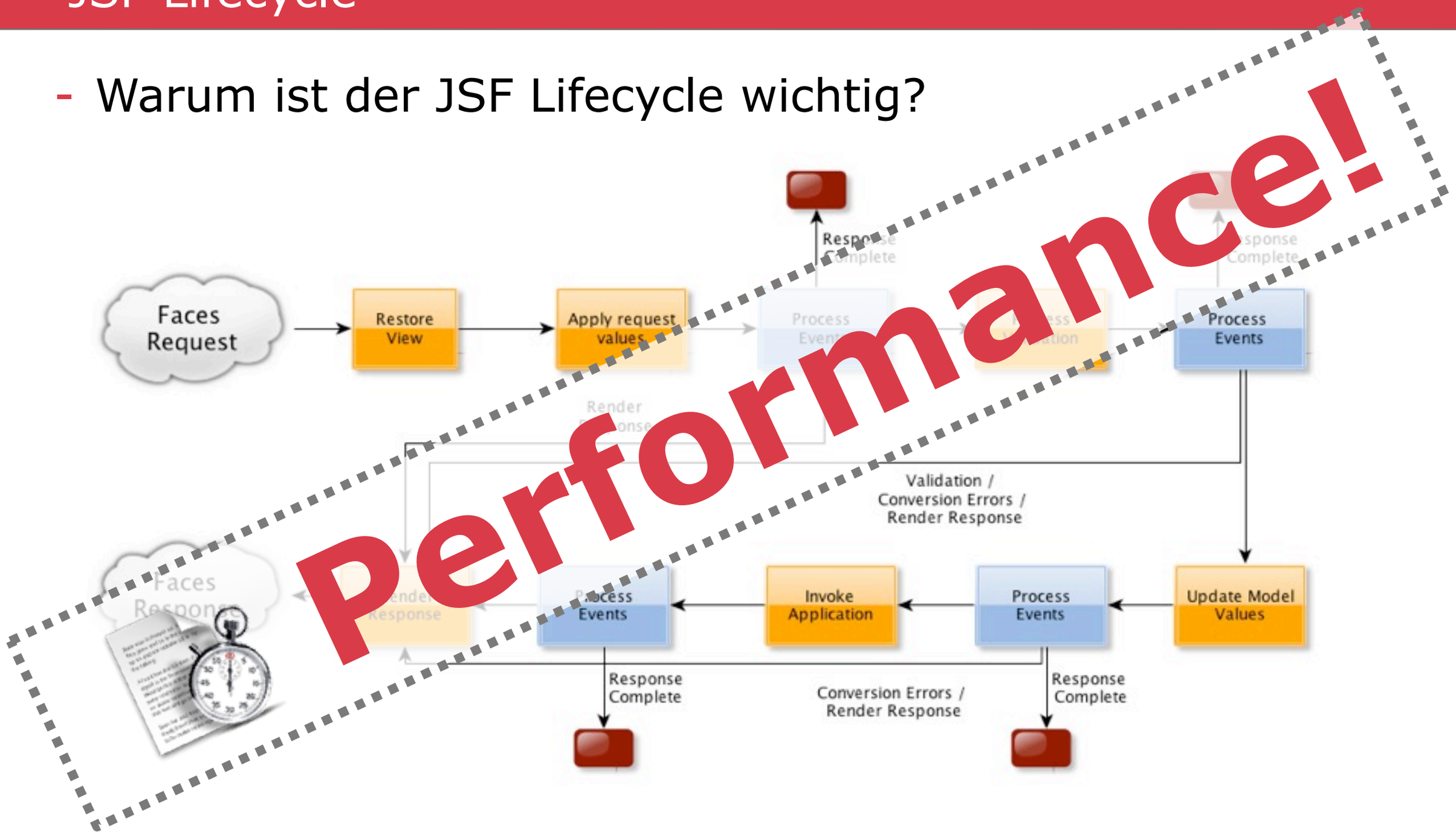
JSF Lifecycle – Render Response



- Zusammenstelle der Rückantwort an den Browser

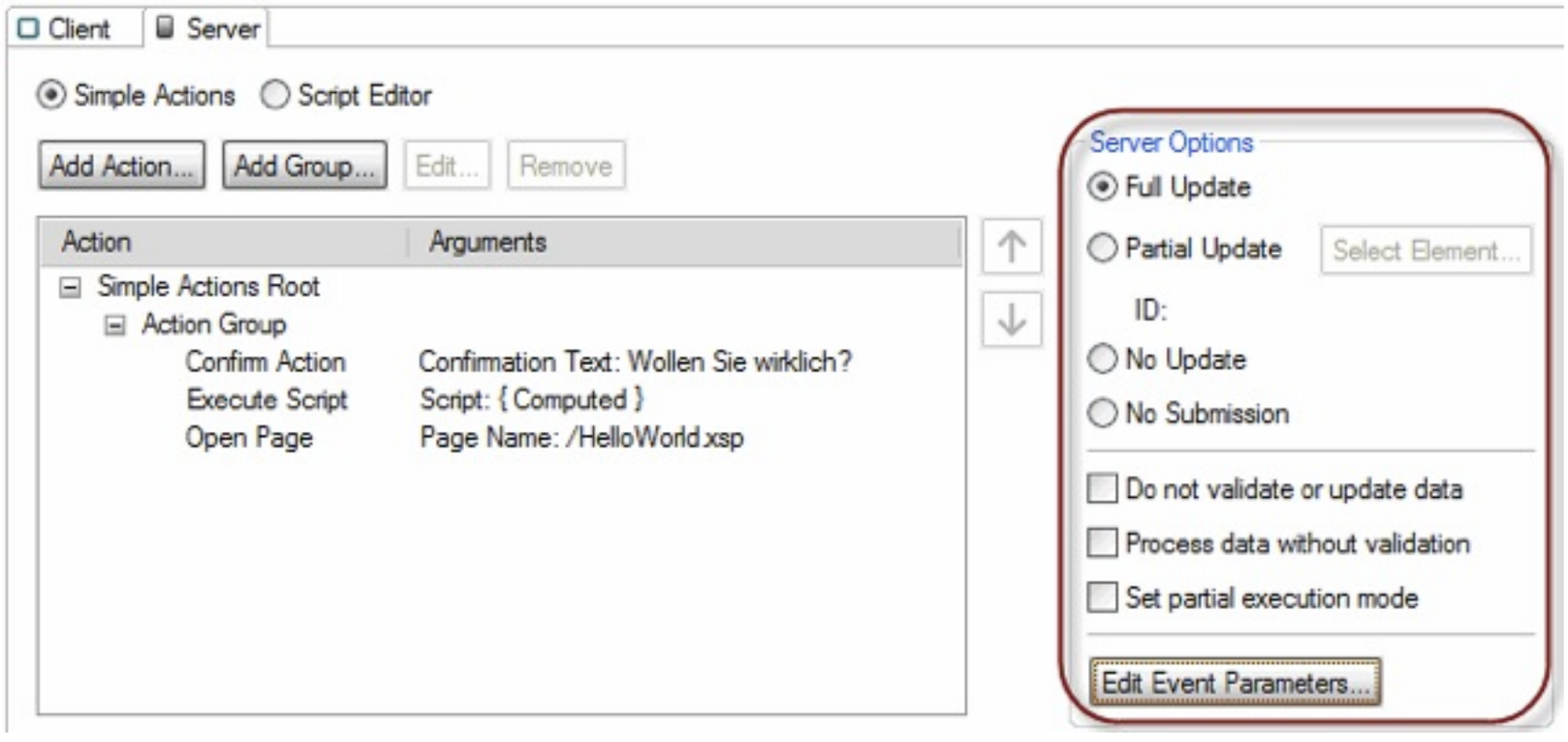
JSF Lifecycle

- Warum ist der JSF Lifecycle wichtig?



JSF Lifecycle - Performance

- SSJS wird während des Lifecycles mehrfach berechnet!
- Server Options bei Events steuern den Lifecycle



«XPage»
Phase

Managed Bean

- Einfache Java-Klasse mit drei Grundregeln
  - Parameterloser Konstruktor
  - Getter- und Setter-Methoden für Datenzugriff
  - Implementiert das Serializable Interface
- Muss in faces-config.xml registriert werden
  - Im Ordner WebContent/WEB-INF

```
<managed-bean>
  <managed-bean-name>myFirstBean</managed-bean-name>
     <managed-bean-class>
       de.assono.ec13.MyFirstBean
     </managed-bean-class>
   <managed-bean-scope>view</managed-bean-scope>
</managed-bean>
```

Managed Bean - Scope

- Managed Bean Scope Optionen
  - application
  - session
  - view
  - request
  - none
- Bei der ersten Verwendung wird die Bean automatisch erzeugt und im Scope abgelegt

NOTES & DOMINO ENTWICKLERCAMP

assono

Managed Bean - Verwendung

- Verwendung der Getter und Setter durch Expression Language (EL)

```
<xp:inputText
 id="inputText1"
 value="#{myFirstBean.myName}">
</xp:inputText>
```

- Verwendung der Methoden durch Aufruf

```
<xp:this.action><!
[CDATA[#{javascript:myFirstBean.doSomething("Hallo
EntwicklerCamp")}]]></xp:this.action>
```

«XPage»
SimpleBean

NOTES & DOMINO ENTWICKLERCAMP

assono

Nützliche JSF-Elemente / Begrifflichkeiten

- FacesContext
- ViewRoot
- ValueBinding
- Validator
- Converter
- MethodBinding
- ChangeListener
- PhaseListener

JavaServer™ Faces

JSF

FacesContext

- Kontext der XPage / JSF-Seite
- SSJS-Objekt: facesContext
- Java-Klasse Browser:
  com.ibm.xsp.context.FacesContextEx
- Java-Klasse XPiNC
  com.ibm.xsp.domino.context.DominoFacesContext
- Zugriff über
  FacesContextEx context =
  FacesContextEx.getCurrentInstance();

FacesContext - Anwendungsbeispiel

- Bestimmung des Pfades über den externen Kontext
  context.getExternalContext()

- Relativer Pfad
  [...].getRequest().getContextPath();
  z.B. /web/webtest.nsf

- Relativer Pfad inkl. XPage-Name
  [...].getRequest().getRequestURI();
  z.B. /web/webtest.nsf/urltest.xsp

- Vollständiger Pfad
  [...].getRequest().getRequestURL().toString();
  z.B. http://bht853/web/webtest.nsf/urlTest.xsp

FacesContext - Nachrichten ausgeben

- Globale oder elementbezogene Nachrichten ausgeben

JavaServer™ Faces JSF

- facesContext.addMessage(java.lang.String clientId, javax.faces.application.FacesMessage message);
- FacesMessage message = new FacesMessage(java.lang.String aMessage);
- Globale Nachricht: clientId = null

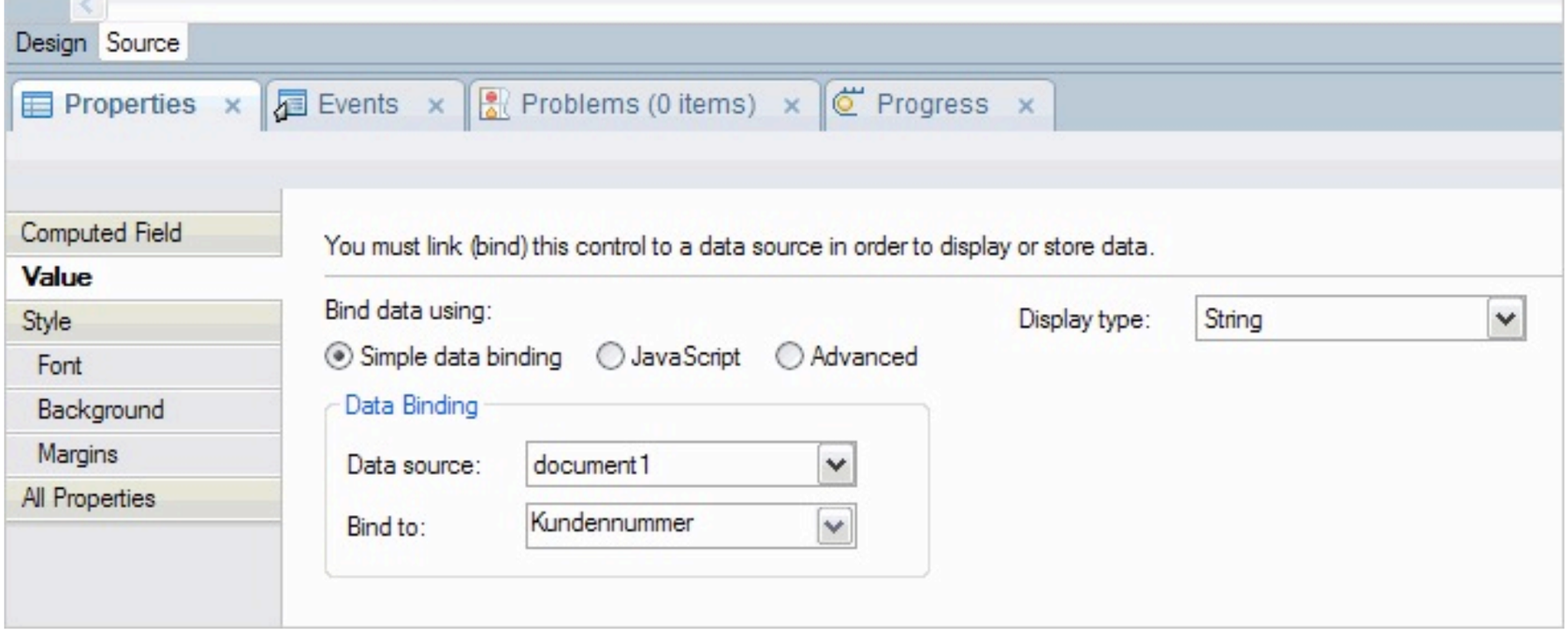
ViewRoot

- Wurzelelement der XPage
- SSJS-Objekt: view
- Java-Klasse: com.ibm.xsp.component.UIViewRootEx2
- Zugriff über
  UIViewRootEx2 viewRoot = (UIViewRootEx2) FacesContext.getCurrentInstance().getViewRoot();
- Anwendungsbeispiel: „Walk the tree“
  List<UIComponent> children = viewRoot.getChildren();

ValueBinding

- Zuweisung einer Eigenschaft eines Seitenelementes


```
38          <xp:td>
39              <xp:text
40                  escape="true"
41                  id="computedField1"
42                  value="#{document1.Kundennummer}">
43              </xp:text>
```



ValueBinding

- Auslesen über UIComponent
  javax.faces.component.UIComponent

- javax.faces.el.ValueBinding vb =
  component.getValueBinding("value");
  String expression = vb.getExpressionString();

- Setzen über Application
  javax.faces.application.Application application =
  facesConfig.getApplication();
  String value="#document1.subject";
  ValueBinding binding =
  application.createValueBinding(value);
  component.setValueBinding("value", binding);

Validator

- Abgeleitet von AbstractValidator
  com.ibm.xsp.validator.AbstractValidator
- Implementieren primär die serverseitige Validierung
- Für clientseitige Validierung muss zusätzlich das Interface ClientSideValidator implementiert werden
  com.ibm.xsp.validator.ClientSideValidator
- Validatoren können an Input-Elemente angefügt werden
  javax.faces.component.UIInput uiInput.addValidator
- Können auch über die faces-config.xml registriert werden

Converter

- Alle Input-Elemente im Browser verwenden Strings
- Konvertieren die Werte der angebundenen Datenquellen von den nativen Datentypen zu Strings und zurück

- Beispiel Konvertierung von Datumswerten
  - vom Browser kommt die Angabe 2013-03-12

```
<xp:inputText id="gueltigAb1" value="#{document1.GueltigAb}"
                        defaultValue="#{javascript:@Today();}">
    <xp:this.converter>
        <xp:convertDateTime type="date" dateStyle="short">
        </xp:convertDateTime>
     </xp:this.converter>
     <xp:dateTimeHelper id="dateTimeHelper1" />
</xp:inputText>
```

MethodBinding

- Methodenaufruf, um es an ein bestimmtes Event zu binden
  javax.faces.el.MethodBinding

- Angabe in Form eines Expression Language (EL) Ausdruckes

- Optional Angabe von Parametern

- Erzeugen über Application

```
Methodbinding mb = FacesContext.getCurrentInstance()
.getApplication().createMethodBinding(
    "#{javascript:thisController.doProcessAfterPageLoad();}",
                        null);
viewRoot.setAfterPageLoad(mb);
```

MethodBinding

- Methodenaufruf, um es an ein bestimmtes Event zu binden
- Angabe in Form eines Expression Language (EL) Ausdruckes

- Optional Angabe von Parametern
- Erzeugen über Application

```
Methodbinding mb = FacesContext.getCurrentInstance()
.getApplication().createMethodBinding(
    "#{javascript:thisController.doProcessAfterPageLoad();}",
                        null);
viewRoot.setAfterPageLoad(mb);
```

ChangeListener

- Werden bei einer Änderung benachrichtigt
- z.B. Interface ValueChangeListener für Änderung eines Wertes

- Methode processValueChange
  void processValueChange(ValueChangeEvent event)
- ValueChangeEvent liefert Werte über
  - getOldValue
  - getNewValue

PhaseListener

- Implementiert das Interface javax.faces.event.PhaseListener
- Primär zwei Methoden
  - beforePhase()
  - afterPhase()
- Muss in faces-config.xml registriert werden
  - Im Ordner WebContent/WEB-INF

```
<lifecycle>
    <phase-istener>
       de.assono.ec13.SimpleLifeCycleListener
    </phase-listener>
</lifecycle>
```

«XPage»
JSFElements

Serializable

- Java Objekte in Scope-Variablen müssen das Interface Serializable (java.io.Serializable) implementieren
  - Für „Keep pages on disk (best scalability)
- Nur Membervariablen, deren Klassen ebenfalls Serializable implementieren, lassen sich „serialisieren“
  - Keine lotus.domino.* Objekte !!!
- Meistens keine Methodenimplementierung notwend

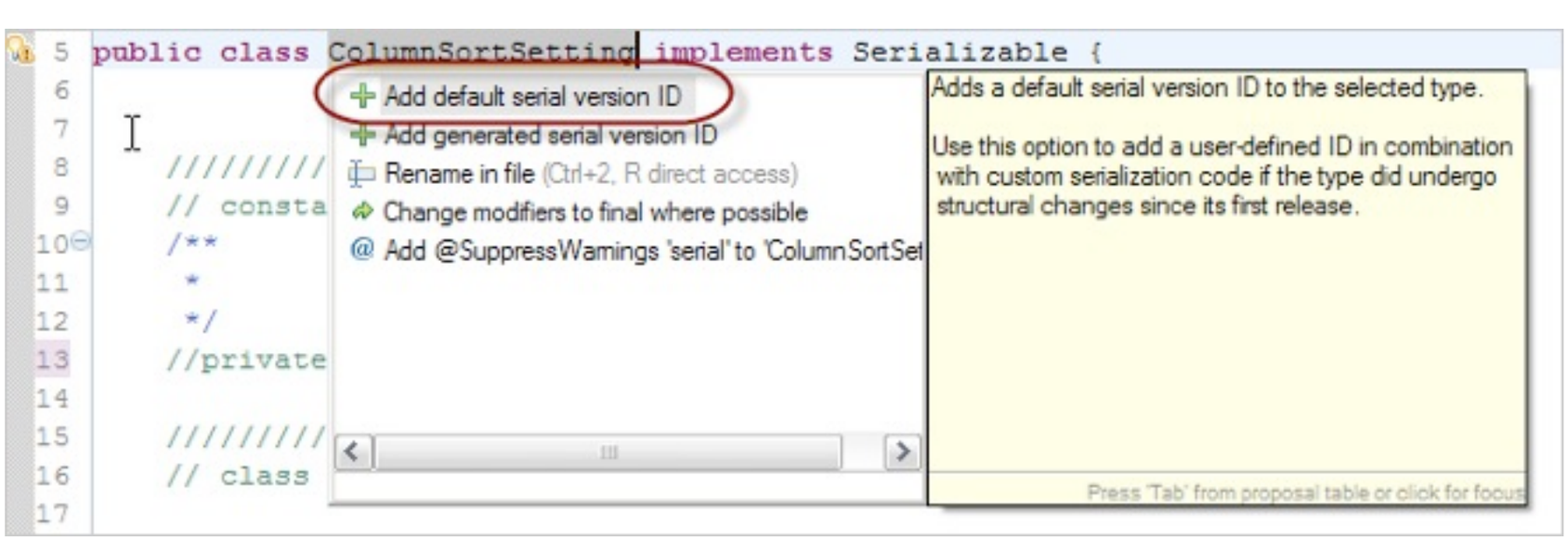
- Galileo: Java ist auch eine Insel
  http://openbook.galileodesign.de/javainsel8/javainsel_14_012.htm

NOTES & DOMINO ENTWICKLERCAMP

assono

Serializable - serialVersionUID

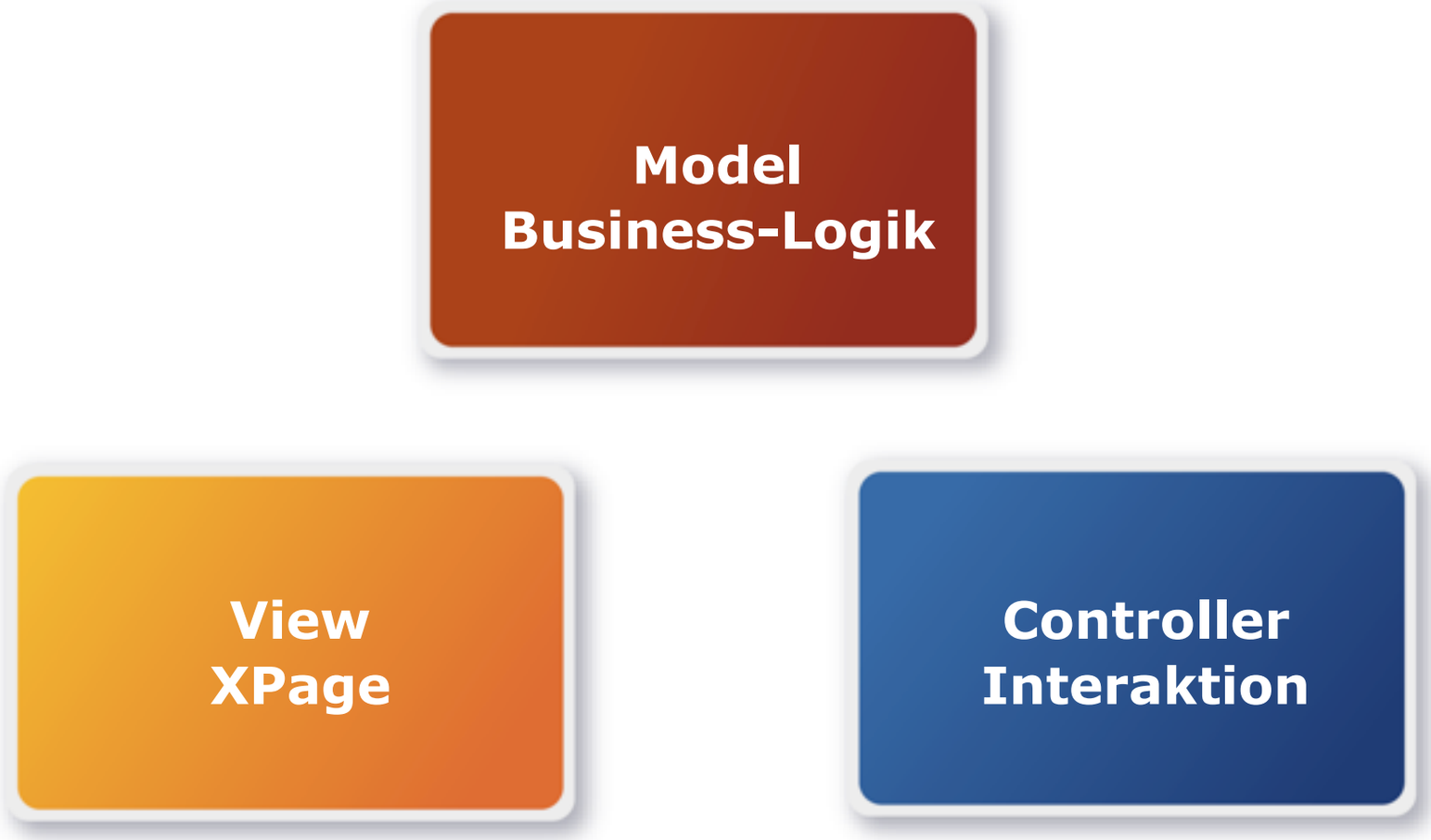
- Warning „The serializable class {Classname} does not declare a static final serialVersionUID field of type long“
- Quick-Fix: Add default serial version ID



- Die generierte Version ist nur bei längerfristiger Persistenz notwendig, um eventuell unterschiedliche Versionen einer Klasse zu unterscheiden

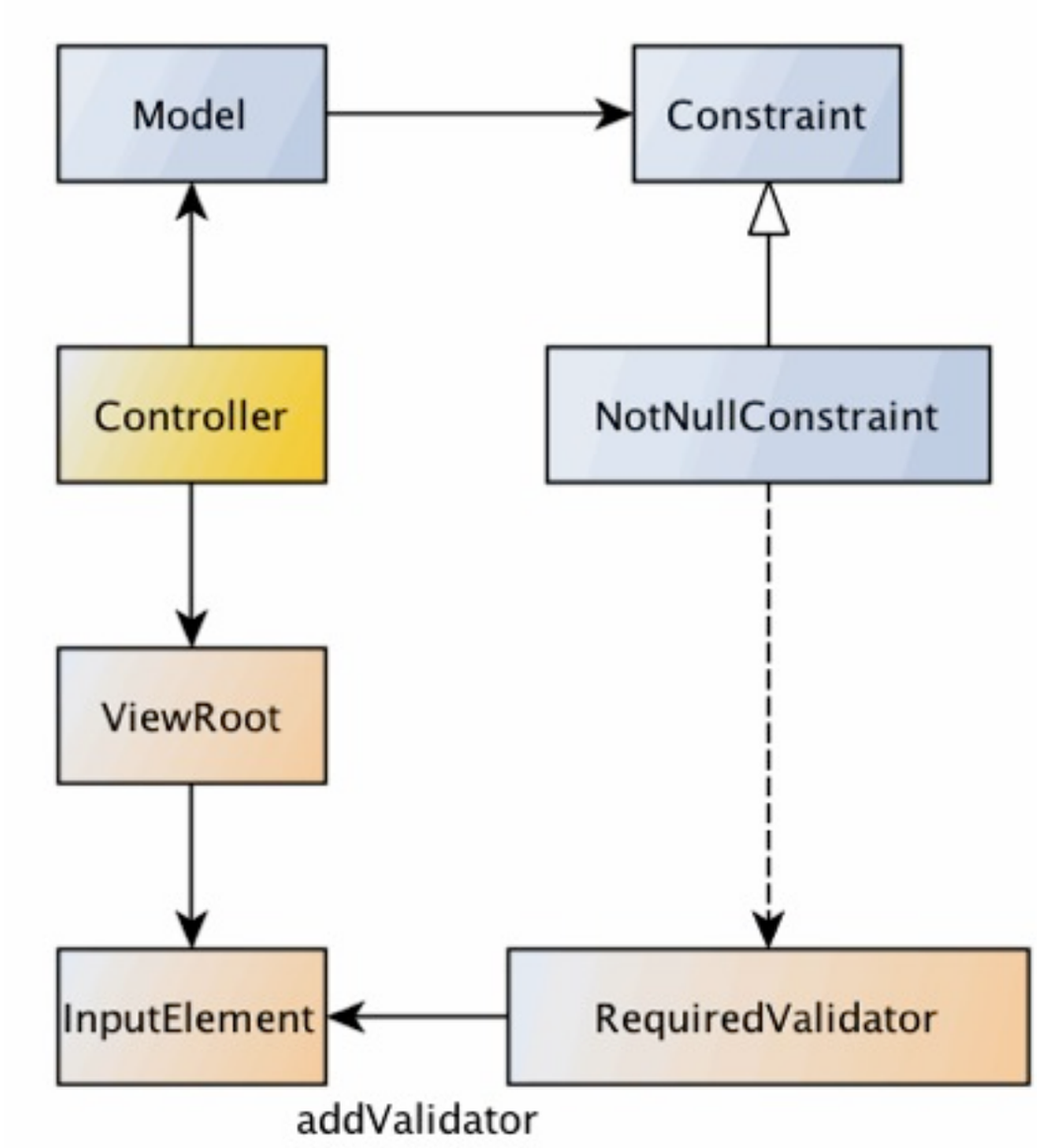
Model-View-Controller

- Trennung zwischen GUI und Business-Logik
- Keine XPages-Klassen im Model

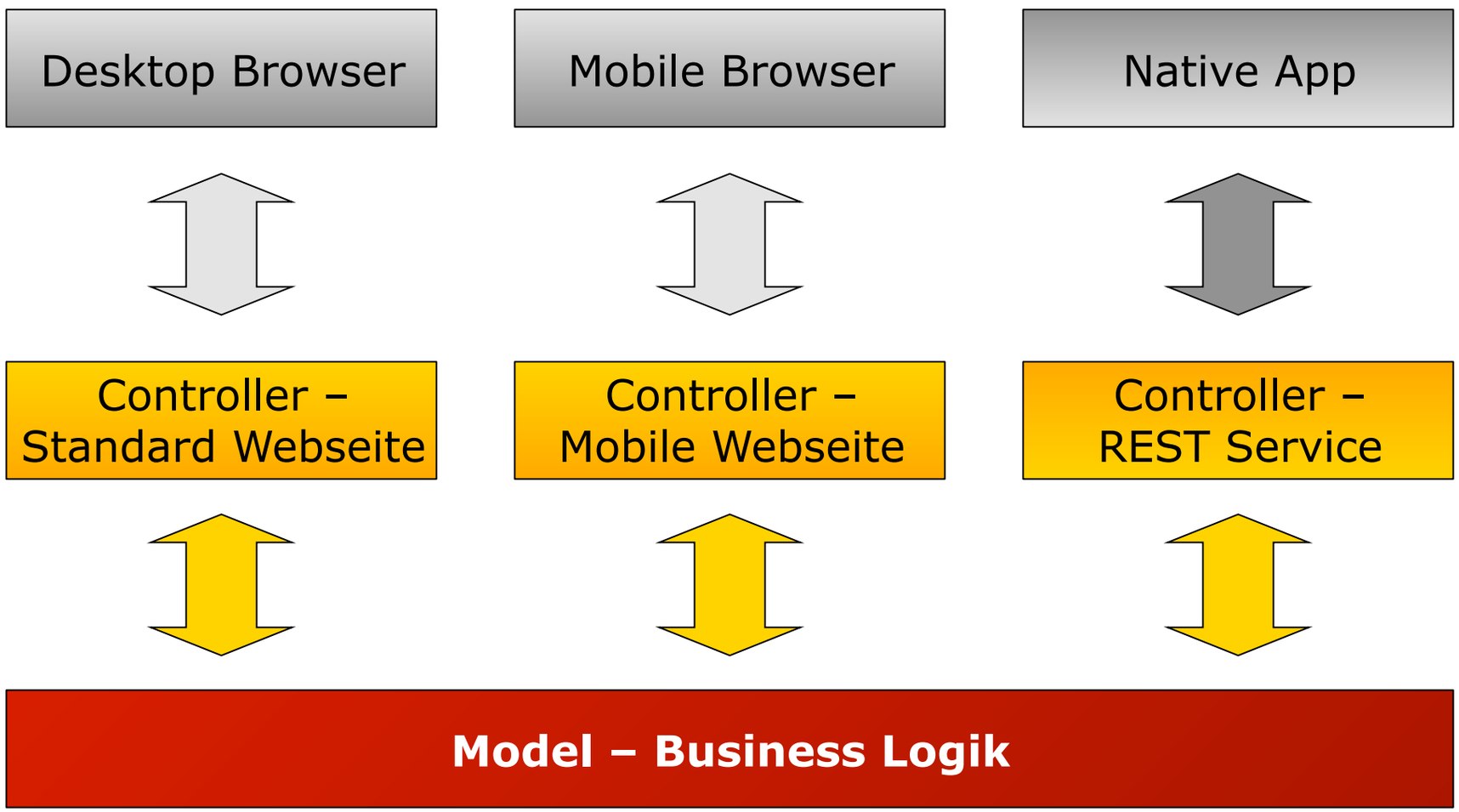


Dynamische Zuweisung Validatoren

- Definition Pflichtfelder als Constraints im Model
- Finder aller Inputelemente der zugehörigen Datenquelle anhand des ValueBindings
- Dynamische Zuordnung von Validatoren bei den Pflichtfeldern



Verwendung mehrere Controller



Unabhängig von dem verwendeten Browser / Gerät wird immer der gleiche Code für die Business Logik verwendet.

NOTES & DOMINO ENTWICKLERCAMP

assono

Quellen

Mindoo Blog – Karsten Lehmann & Tammo Riedinger

http://blog.mindoo.com

Java & XPages Artikel

Sprecher auf dem EntwicklerCamp!

Ressourcen

- XPages Extensibility API Documentation
  http://www-10.lotus.com/ldd/ddwiki.nsf/dx/xpages_extensibility_api_documentation
  JavaDoc aller XPages Klassen

- Declan Sciolla-Lynch: XPage Java Roots
  http://www.qtzar.com/xpage-java-roots/

- jNotes: XPages und Managed Beans
  http://www.jnotes.de/clients/jnotes/CMS2Content.nsf/content/post.html?Open&postid=OBUE-8ALGKR

- Tim Tripcony: Taking the scary out of Java in XPages
  http://www.timtripcony.com/blog.nsf/d6plinks/TTRY-95B6KS

Bücher

Thomas Ekert:

„Java unter Lotus Domino“

Springer, ISBN 978-3540221760, 804 S.

(Jahrgang 2006, Deutsch!, Preis 9,99€!)

Web-Seite: http://www.springer.com/computer/swe/book/978-3-540-22176-0

NOTES & DOMINO ENTWICKLERCAMP

assono

Fragen?

jetzt stellen – oder später:

bhort@assono.de

http://www.assono.de/blog

040/73 44 28-315

assono
IT-Consulting & Solutions

Folien & Beispieldatenbank unter
http://www.assono.de/blog/d6plinks/EntwicklerCamp-2013-XPages-Java