NOTES & DOMINO
ENTWICKLERCAMP

assono

Java Managed Beans

EntwicklerCamp
03. März 2015

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/IBM Connect 2014/IBM ConnectED 2015

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

assono
IT-Consulting & Solutions

NOTES & DOMINO ENTWICKLERCAMP

assono

Agenda

- Motivation
- Managed Beans
- Einbindung in XPages
- Einsatzszenarien
- Trennung zwischen Business Logik und Darstellung
- Fragen & Antworten

XPages

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.ec15.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

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>
```

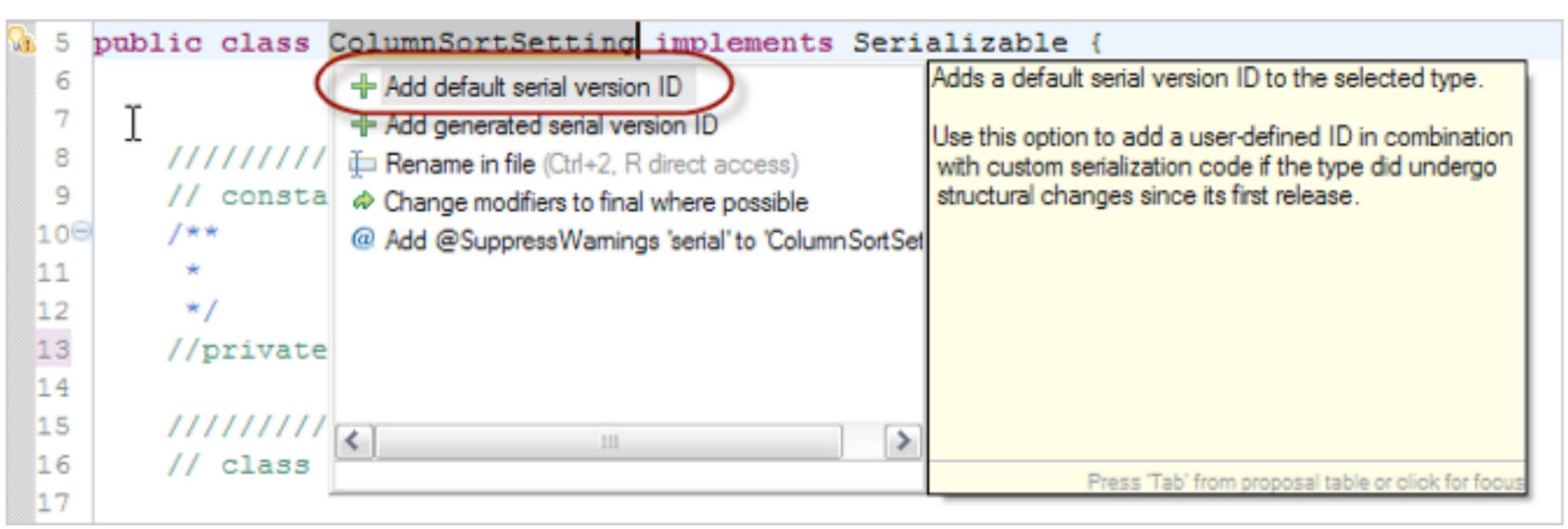
«XPage»
SimpleBean

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

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

Informationsaustausch zwischen Beans

- Zugriff auf die HashMap in den Scope-Variablen
- Zugriff auf die Beans via
  - VariableResolver
  - ValueBinding
- Bean Injection und Managed Properties
  - Übergabe einer Bean als Property der anderen Bean
  - Definition in der faces-config.xml

NOTES & DOMINO ENTWICKLERCAMP

assono

Zugriff auf die HashMap der Scope Variablen

- Über den FacesContext Zugriff auf den ExternalContext und darüber Zugriff auf die Map

```
public static Map<String, Object> getSessionScope() {
      FacesContext ctx = FacesContext.getCurrentInstance();
      if (ctx != null) {
         return ctx.getExternalContext().getSessionMap();
      }
      return null;
   }
```

JavaServer™ Faces JSF

NOTES & DOMINO ENTWICKLERCAMP

assono

Zugriff auf „globale“ Objekte via Variable Resolver

- Über den FacesContext Zugriff auf die Application und darüber Zugriff auf den VariableResolver

```
public static MyFirstBean getMyFirstBean() {
      FacesContext facesContext = FacesContext.getCurrentInstance();
      MyFirstBean bean = (MyFirstBean) facesContext.getApplication()
            .getVariableResolver().resolveVariable(facesContext,
                  "myFirstBean");
      return bean;

}
```

JavaServer™ Faces JSF

NOTES & DOMINO ENTWICKLERCAMP

assono

Zugriff auf „globale“ Objekte via Variable Resolver

- Über den FacesContext Zugriff auf die Application und ein ValueBinding erzeugt

```
public static MyFirstBean getMyFirstBeanViaValueBinding() {
      FacesContext facesContext = FacesContext.getCurrentInstance();
      MyFirstBean bean = (MyFirstBean) facesContext.getApplication()
            .createValueBinding("#{myFirstBean}").getValue(facesContext);
      return bean;
}
```

JavaServer™ Faces JSF

Managed Properties

- Über faces-config.xml können Properties gesetzt werden

```
<managed-bean>
   <managed-bean-name>demoBean</managed-bean-name>
   <managed-bean-class>de.assono.ec15.DemoBean</managed-bean-class>
   <managed-bean-scope>view</managed-bean-scope>
   <managed-property>
      <property-name>title</property-name>
      <value>EntwicklerCamp 2015</value>
   </managed-property>
</managed-bean>
```

Bean Injection

- Als Property kann eine andere Bean übergeben werden

```
<managed-bean>
   <managed-bean-name>configurationBean</managed-bean-name>
   <managed-bean-class>de.assono.ec15.ConfigurationBean</managed-bean-class>
   <managed-bean-scope>application</managed-bean-scope>
</managed-bean>

<managed-bean>
   <managed-bean-name>demoControllerBean</managed-bean-name>
   <managed-bean-class>de.assono.ec15.DemoController</managed-bean-class>
   <managed-bean-scope>view</managed-bean-scope>
   <managed-property>
      <property-name>configuration</property-name>
      <value>#{configurationBean}</value>
   </managed-property>
</managed-bean>
```

«XPage»
BeanCommunication

NOTES & DOMINO ENTWICKLERCAMP

assono

Einsatzszenarien

- Konfiguration
- Kapselung häufig genutzter Funktionalität
- Controller für Model-View-Controller

Konfiguration

- Vorteil: Werte nur einmal auslesen
  - Werte aus Profil-Dokument oder normalen Konfigurations-Dokumenten nur einmal auslesen
  - Werte in einer HashMap speichern
- Nachteil: Werte werden nur einmal ausgelesen
  - Mechanismus für erneutes Auslesen vorsehen

NOTES & DOMINO ENTWICKLERCAMP

assono

Profile-Dokumente in XPages Anwendungen

- Profile-Dokumente werden im Speicher des Serves gehalten
- Wird ein Profil-Dokument über Notes Client geändert, wird die Änderung nicht unmittelbar sichtbar
- Alternativ Schattendokumente
  - Beim Speichern des Profil-Dokumentes alle Werte in ein reguläres Dokument speichern
  - Im Profil-Dokument UNID des Schattendokumentes hinterlegen
  - Schneller Zugriff auf Profil-Dokument zum Auslesen der UNID

«XPage»
Configuration

UserBean

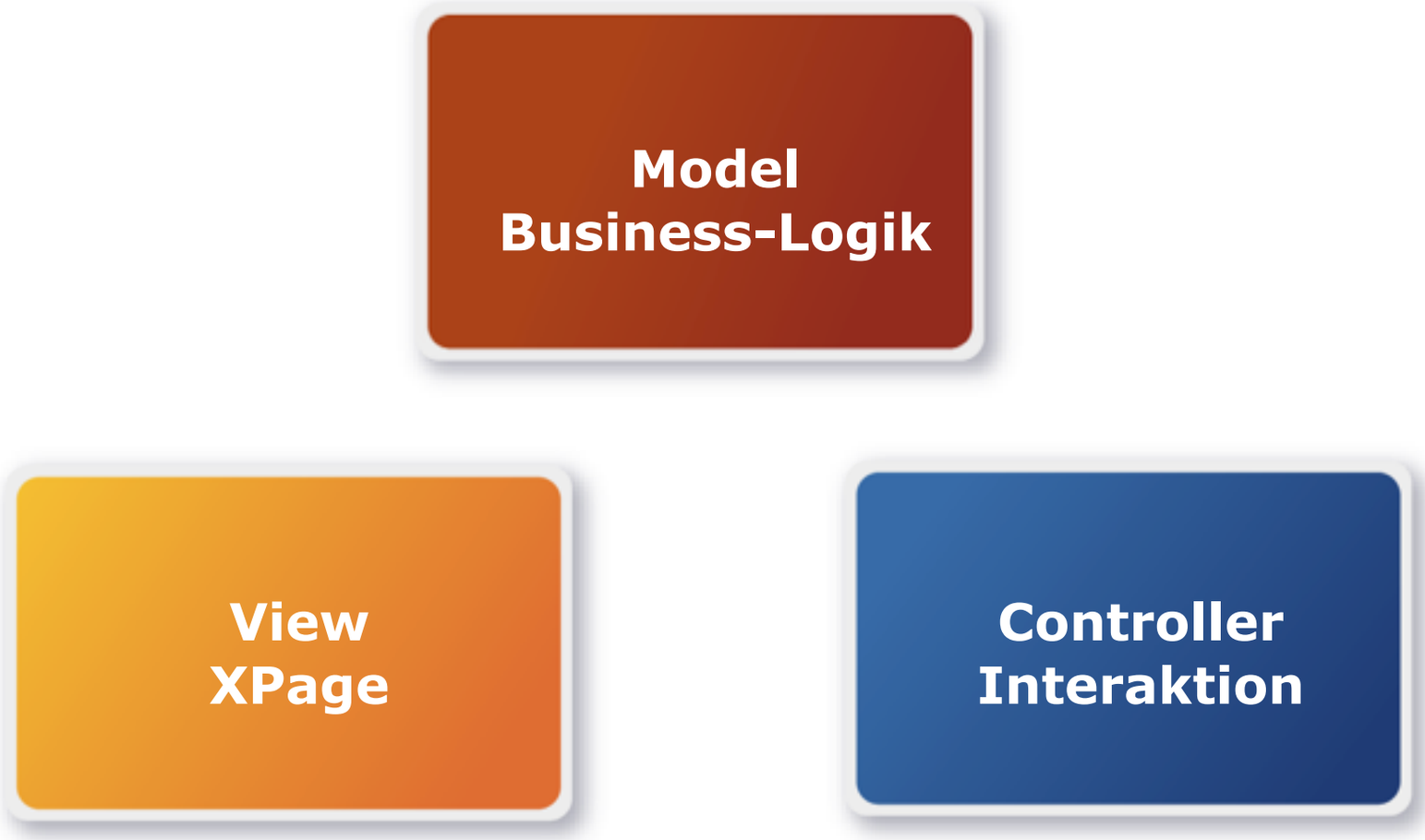
- Beispiel User-Bean aus der Extension Library

Property	Erklärung
`userBean.commonName`	Vorname und Nachname
`userBean.id`	Notes Name, wenn Domino die Quelle ist
`userBean.canonicalName`	Notes Name
`userBean.abbreviatedName`	Notes Name, abgekürzte Form
`userBean.thumbnailUrl`	URL zum Avatar-Bild
`userBean.canCreateDocs`	Aktueller Anwender darf Dokumente erstellen
`userBean.canDeleteDocs`	Aktueller Anwender darf Dokumente löschen
`userBean.accessLevel`	Zugriffsrecht als Zahl von 0=Kein Zugriff bis 6=Manager
`userBean.accessLevelAsString`	Zugriffsrecht als Text, z.B. MANAGER

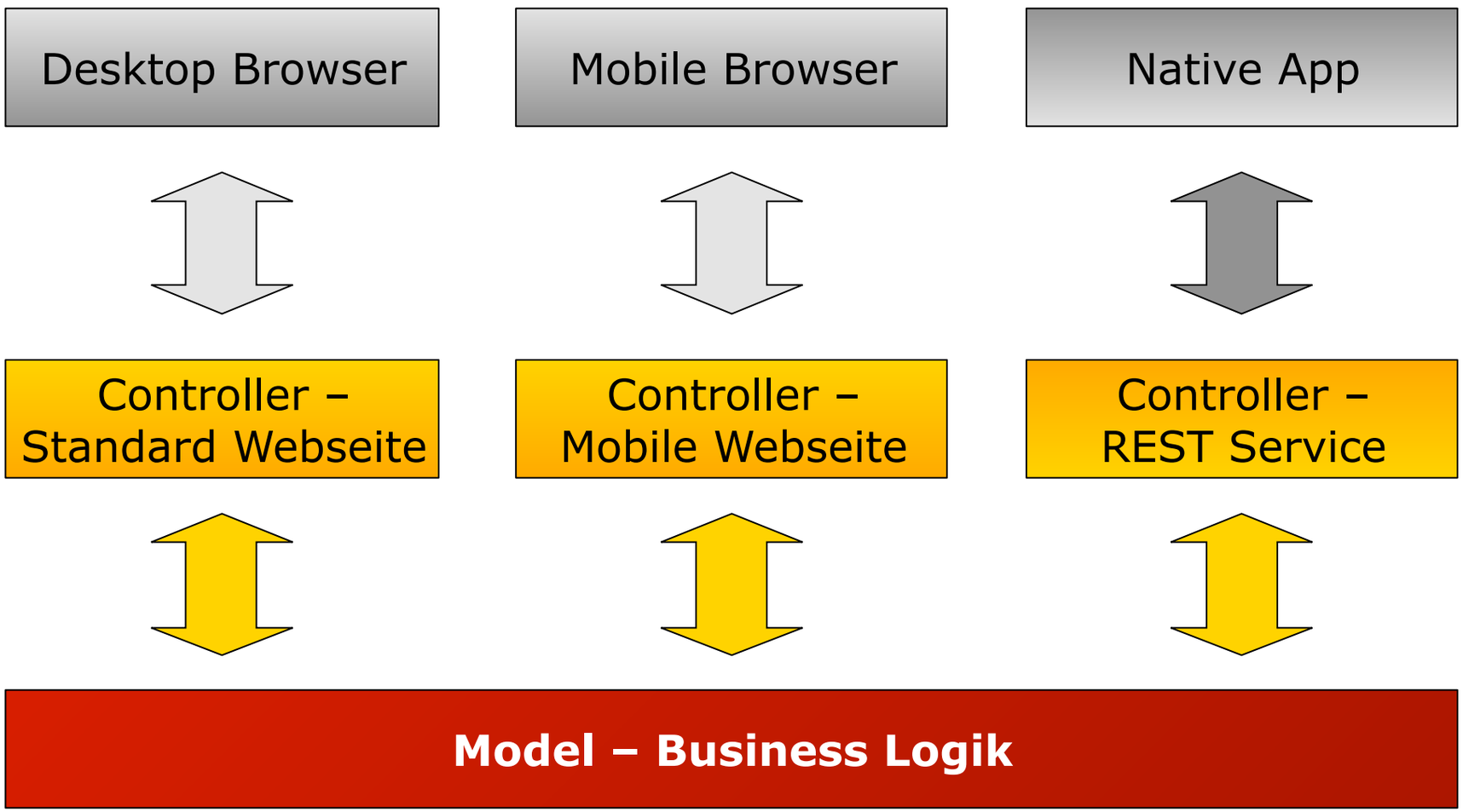
```
<xp:button
   loaded="${userBean.canCreateDocs}">
</xp:button>
```

Model-View-Controller

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



Verwendung mehrere Controller



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

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-2015-Java-Managed-Beans