
L8. TUTORIAL Eclipse [III]

JSF: Form Complex Master/Detail

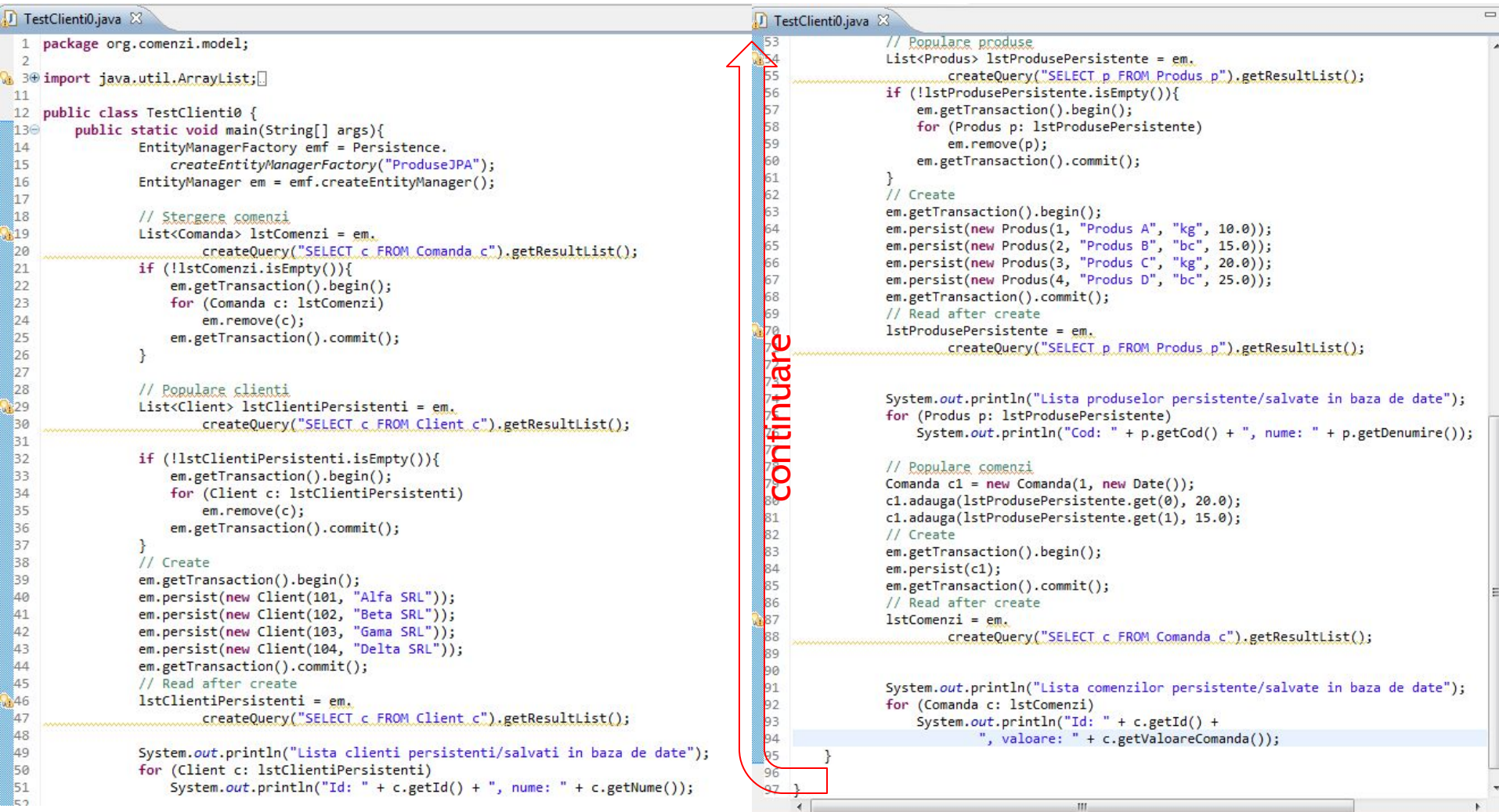
IDE: Eclipse Oxygene JEE [4.7]
Distribution: **OEPE 12.2.1.6.x**
Implementare JSF: **JSF 2.2 / Apache MyFaces 2.2.x**
Implementare JPA: JPA 2.1 / EclipseLink 2.7.0
Server Web: **Apache Tomcat 8.5**

SGBD: PostgreSQL9/PostgreSQL10
Driver JDBC: postgresql-42.x.jdbc

Plan

- 1. Implementarea modelului de date master/detail
 - 2. Crearea formularului principal
 - configurare rubrici;
 - codificare acțiuni;
 - invocare acțiuni prin butoane de comandă.
 - 3. Crearea sub-formularului detail
 - crearea și configurarea gridului cu detalii;
 - controlul selectării unui element specific din grid;
 - acțiuni tranzacționale specifice sub formularului
-

Exemplu TestJPA simplificat pentru popularea structurilor relaționale (clicenți, produse, comenzi)



```
1 package org.comenzi.model;
2
3 import java.util.ArrayList;
4
11
12 public class TestClienti0 {
13     public static void main(String[] args){
14         EntityManagerFactory emf = Persistence.
15             createEntityManagerFactory("ProduseJPA");
16         EntityManager em = emf.createEntityManager();
17
18         // Stergere comenzi
19         List<Comanda> lstComenzi = em.
20             createQuery("SELECT c FROM Comanda c").getResultList();
21         if (!lstComenzi.isEmpty()){
22             em.getTransaction().begin();
23             for (Comanda c: lstComenzi)
24                 em.remove(c);
25             em.getTransaction().commit();
26         }
27
28         // Populare clicenți
29         List<Client> lstClientiPersistenți = em.
30             createQuery("SELECT c FROM Client c").getResultList();
31
32         if (!lstClientiPersistenți.isEmpty()){
33             em.getTransaction().begin();
34             for (Client c: lstClientiPersistenți)
35                 em.remove(c);
36             em.getTransaction().commit();
37         }
38         // Create
39         em.getTransaction().begin();
40         em.persist(new Client(101, "Alfa SRL"));
41         em.persist(new Client(102, "Beta SRL"));
42         em.persist(new Client(103, "Gama SRL"));
43         em.persist(new Client(104, "Delta SRL"));
44         em.getTransaction().commit();
45         // Read after create
46         lstClientiPersistenți = em.
47             createQuery("SELECT c FROM Client c").getResultList();
48
49         System.out.println("Lista clicenți persistenți/salvați în baza de date");
50         for (Client c: lstClientiPersistenți)
51             System.out.println("Id: " + c.getId() + ", nume: " + c.getNume());
52     }
53 }
54
55 // Populare produse
56 List<Produs> lstProdusePersistente = em.
57     createQuery("SELECT p FROM Produs p").getResultList();
58 if (!lstProdusePersistente.isEmpty()){
59     em.getTransaction().begin();
60     for (Produs p: lstProdusePersistente)
61         em.remove(p);
62     em.getTransaction().commit();
63 }
64 // Create
65 em.getTransaction().begin();
66 em.persist(new Produs(1, "Produs A", "kg", 10.0));
67 em.persist(new Produs(2, "Produs B", "bc", 15.0));
68 em.persist(new Produs(3, "Produs C", "kg", 20.0));
69 em.persist(new Produs(4, "Produs D", "bc", 25.0));
70 em.getTransaction().commit();
71 // Read after create
72 lstProdusePersistente = em.
73     createQuery("SELECT p FROM Produs p").getResultList();
74
75 System.out.println("Lista produselor persistente/salvate în baza de date");
76 for (Produs p: lstProdusePersistente)
77     System.out.println("Cod: " + p.getCod() + ", nume: " + p.getDenumire());
78
79 // Populare comenzi
80 Comanda c1 = new Comanda(1, new Date());
81 c1.adauga(lstProdusePersistente.get(0), 20.0);
82 c1.adauga(lstProdusePersistente.get(1), 15.0);
83 // Create
84 em.getTransaction().begin();
85 em.persist(c1);
86 em.getTransaction().commit();
87 // Read after create
88 lstComenzi = em.
89     createQuery("SELECT c FROM Comanda c").getResultList();
90
91 System.out.println("Lista comenzilor persistente/salvate în baza de date");
92 for (Comanda c: lstComenzi)
93     System.out.println("Id: " + c.getId() +
94         ", valoare: " + c.getValoareComanda());
95 }
96 }
97 }
```

1. Implementarea modelului de date

- 1.1 Identificarea asocierii 1-M din modelul afacerii (proiectul JPA)
 - 1.2 Declararea structurilor care vor forma modelul de date M/D (proiectul JSF)
 - declararea modelului prin proprietățile clasei suport ;
 - expunerea modelului de date (getterii și setterii).
 - 1.3 Declararea obiectului managed-bean – suport al formularului grafic
-

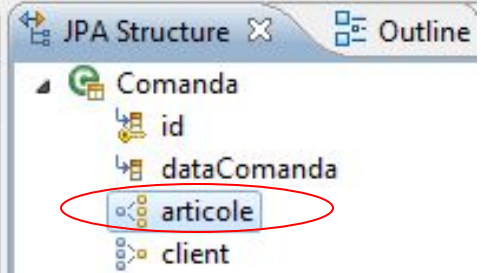
1.1 Identificarea asocierii 1-M din modelul afacerii (proiectul JPA)

- Cele mai simple de configurat formulare M/D au la bază o asociere 1-M în modelul afacerii. Această asociere permite sincronizarea “naturală” a formularului master și sub-formularului detail.
 - Parametrii JPA ai asociației privind salvarea și/sau ștergerea în cascadă *@OneToMany(... cascade=...)* influențează comportamentul de salvare al formularului.
-

1.1 Identificarea asocierii 1-M din modelul afacerii (proiectul JPA)

- În următoarele 2 pagini, atenție la:
 - tipul atributului *comenzi* care implementează “capătul” asociației în clasa *Comanda*;
 - tipul adnotării JPA și valoarea atributului *cascade* al acestei adnotări: constanta *ALL* se referă la faptul că:
 - ștergerea comenzii va antrena și ștergerea articolelor;
 - salvarea comenzii va antrena și salvarea articolelor.
 - tipul atributului *comanda* care implementează capătul asociației în clasa *ArticolComanda*.
-

```
Comanda.java
1 package org.comenzi.model;
2
3 import java.util.ArrayList;
4
14
15 @Entity
16 public class Comanda {
17     @Id
18     private Integer id;
19
20     @Temporal (DATE)
21     private Date dataComanda;
22
23     @OneToMany(mappedBy = "comanda", cascade = ALL)
24     private List<ArticolComanda> articolo
25         = new ArrayList<ArticolComanda>();
26
```



Problems Console JPA Details Servers

Attribute 'articolo' is mapped as one to many.

▼ One to Many

Target entity: Default (org.comenzi.model.ArticolComanda) Browse...

Fetch: Default (Lazy)

Join fetch: <None>

☐ Private owned

☒ Orphan removal (False)

Cascade

☒ All ☐ Persist ☐ Merge ☐ Remove ☐ Refresh ☐ Detach

ArticolComanda.java Comanda.java

```
1 package org.comenzi.model;
2
3 import javax.persistence.Entity;
4
5
6 @Entity
7 public class ArticolComanda {
8     @Id
9     @GeneratedValue(strategy = SEQUENCE)
10     private Integer id;
11
12     @ManyToOne
13     private Produs produs;
14
15     private Double cantitate;
16
17     @ManyToOne
18     private Comanda comanda;
```

JPA Structure

- ArticolComanda
 - id
 - produs
 - cantitate
 - comanda

Problems Console JPA Details Servers

Attribute 'comanda' is mapped as many to one.

Many to One

Target entity: Default (org.comenzi.model.Comanda) Browse...

Fetch: Default (Eager)

Join fetch: <None>

☒ Optional (True)

Cascade

- ☐ All
- ☐ Persist
- ☐ Merge
- ☐ Remove
- ☐ Refresh
- ☐ Detach

1.2 Structurile modelului M-D: master

- Partea *master*:
 - colecția de entități master: *comenzi*, expuse însă ca un Map<String, Comanda> de entități *Comanda* indexate după *id-ul* lor (vezi implementarea operației *getComenzi()* din pagina următoare) gata pregătit pentru a alimenta o listă de selecție-navigare;
 - entitatea master considerată curentă: *comanda* ale cărei valori din proprietăți vor popula formularul principal;
-

Structurile modelului M-D: detail

- Partea *detail*:
 - colecția de entități detail (care vor popula sub-formularul grid) *asociate entității master* se obține indirect: *comanda.getArticole()*, sau ***#formComenzi.comanda.articole***;
 - colecția de entități *produse* pentru alimentarea listei care va expune proprietatea *produs* asociată fiecărei entități detail *ArticolComanda*. Aceasta va fi expusă tot ca un *Map<String, Produs>* de entități *Produs* indexate după codul lor (vezi implementarea *getProduse()* din pagina următoare).
- Modelul va fi accesibil prin proprietățile unui obiect suport *managed-bean* cu numele *formComenzi*.

Declararea și expunerea modelului de date

```
FormComenzi.java X
1 package org.comenzi.forms;
2
3 import java.util.ArrayList;
21
22 public class FormComenzi {
23     // Modelul de date
24     private List<Comanda> comenzi = new ArrayList<Comanda>();
25     private Comanda comanda;
26     private List<Produs> produse = new ArrayList<Produs>();
27
28     public List<Comanda> getComenziList() {
29         return comenzi;
30     }
31     public Map<String, Comanda> getComenzi(){
32         Map<String, Comanda> comenziMap = new HashMap<String, Comanda>();
33         for (Comanda c: this.comenzi)
34             comenziMap.put(c.getId().toString(), c);
35         return comenziMap;
36     }
37     public Comanda getComanda() {
38         return comanda;
39     }
40     public void setComanda(Comanda comanda) {
41         this.comanda = comanda;
42     }
43     public List<Produs> getProduseList() {
44         return produse;
45     }
46     public Map<String, Produs> getProduse(){
47         Map<String, Produs> produseMap = new HashMap<String, Produs>();
48         for (Produs p: this.produse)
49             produseMap.put(p.getDenumire().toString(), p);
50         return produseMap;
51     }
52 }
```

Implementare internă
modelului

Expunerea
modelului

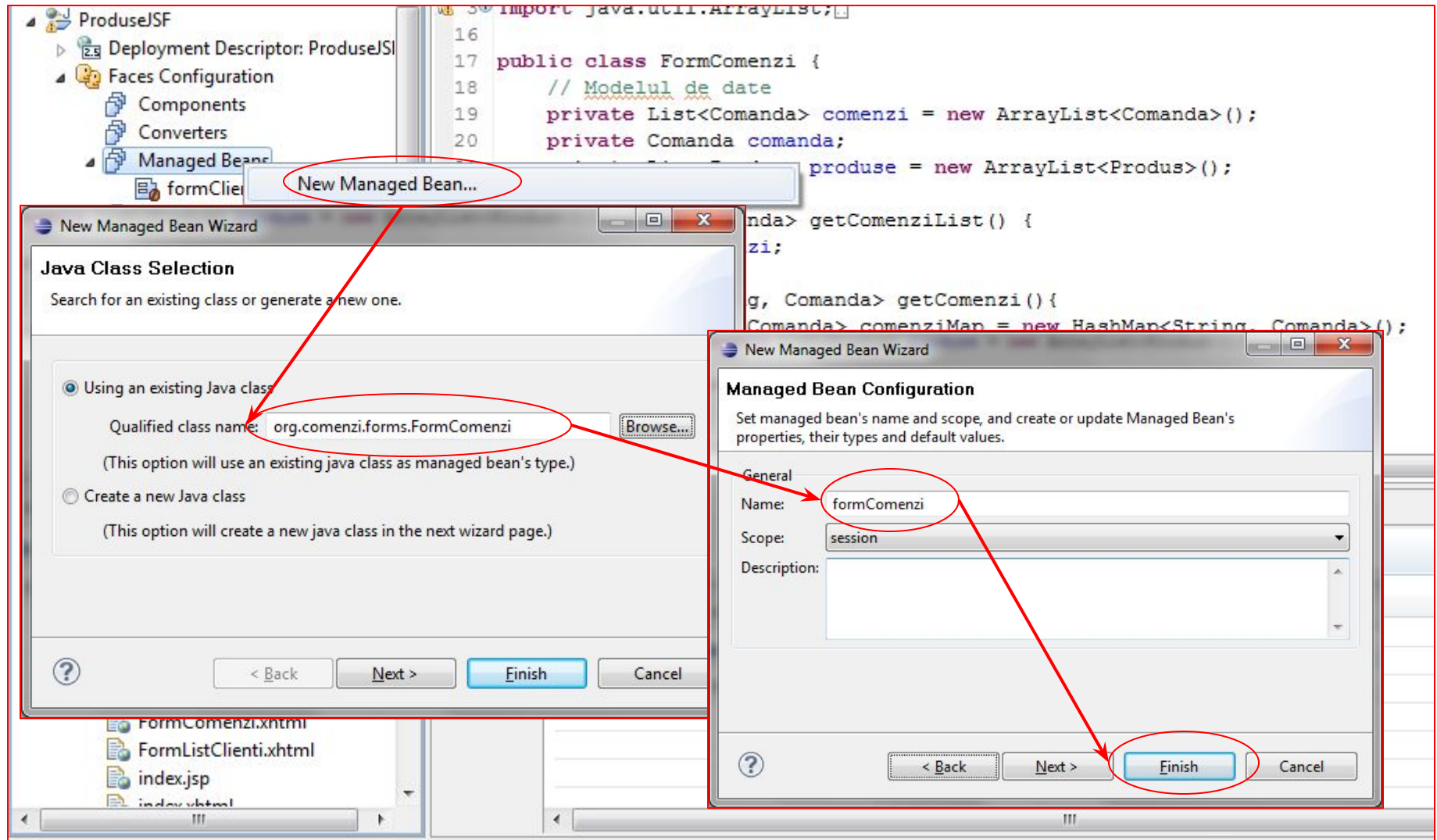
Inițializarea modelului de date

```
FormComenzi.java X
47
48 // Initalizare model
49 private EntityManager em;
50 public FormComenzi() {
51     EntityManagerFactory emf = Persistence.createEntityManagerFactory("ProduseJPA");
52     em = emf.createEntityManager();
53
54     this.comenzi = em.createQuery("SELECT o FROM Comanda o").getResultList();
55     if (!this.comenzi.isEmpty()) {
56         this.comanda = this.comenzi.get(0);
57     }
58
59     this.produse = em.createQuery("SELECT o FROM Produs o").getResultList();
60 }
61
62 public void previousComanda(ActionEvent evt) {
63     Integer idxCurent = this.comenzi.indexOf(this.comanda);
64     if (idxCurent - 1 >= 0)
65         this.comanda = this.comenzi.get(idxCurent - 1);
66 }
67
68 public void nextComanda(ActionEvent evt) {
69     Integer idxCurent = this.comenzi.indexOf(this.comanda);
70     if (idxCurent + 1 < this.comenzi.size())
71         this.comanda = this.comenzi.get(idxCurent + 1);
72 }
```

Inițializarea modelului

Suportul deja pregătit
pentru navigare secvențială

Declararea obiectului *managed-bean formComenzi* care va fi invocat în formularul grafic (în fișierul xhtml)

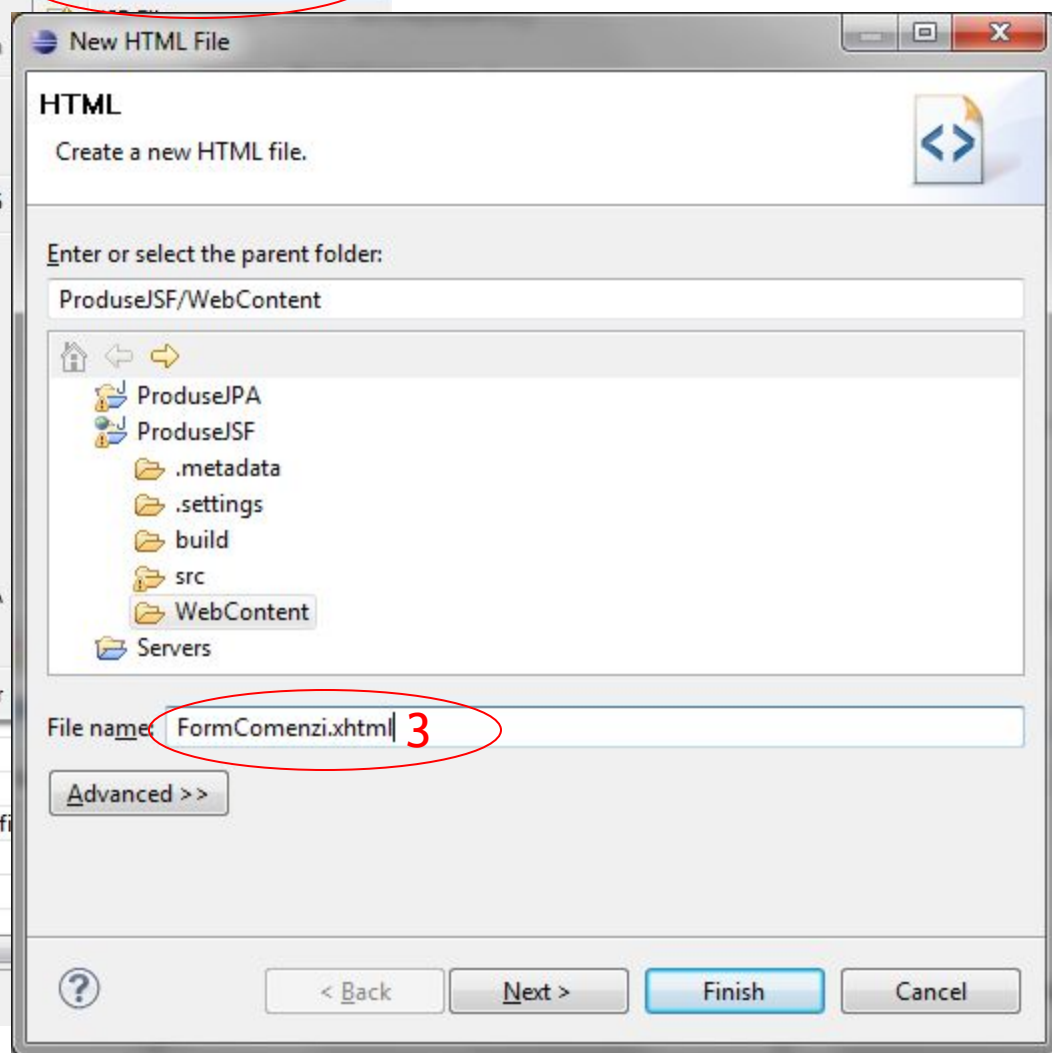
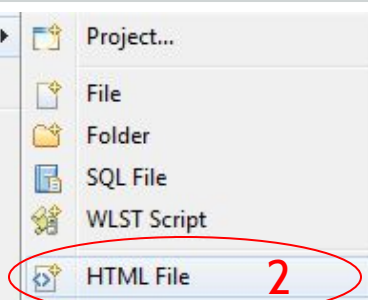
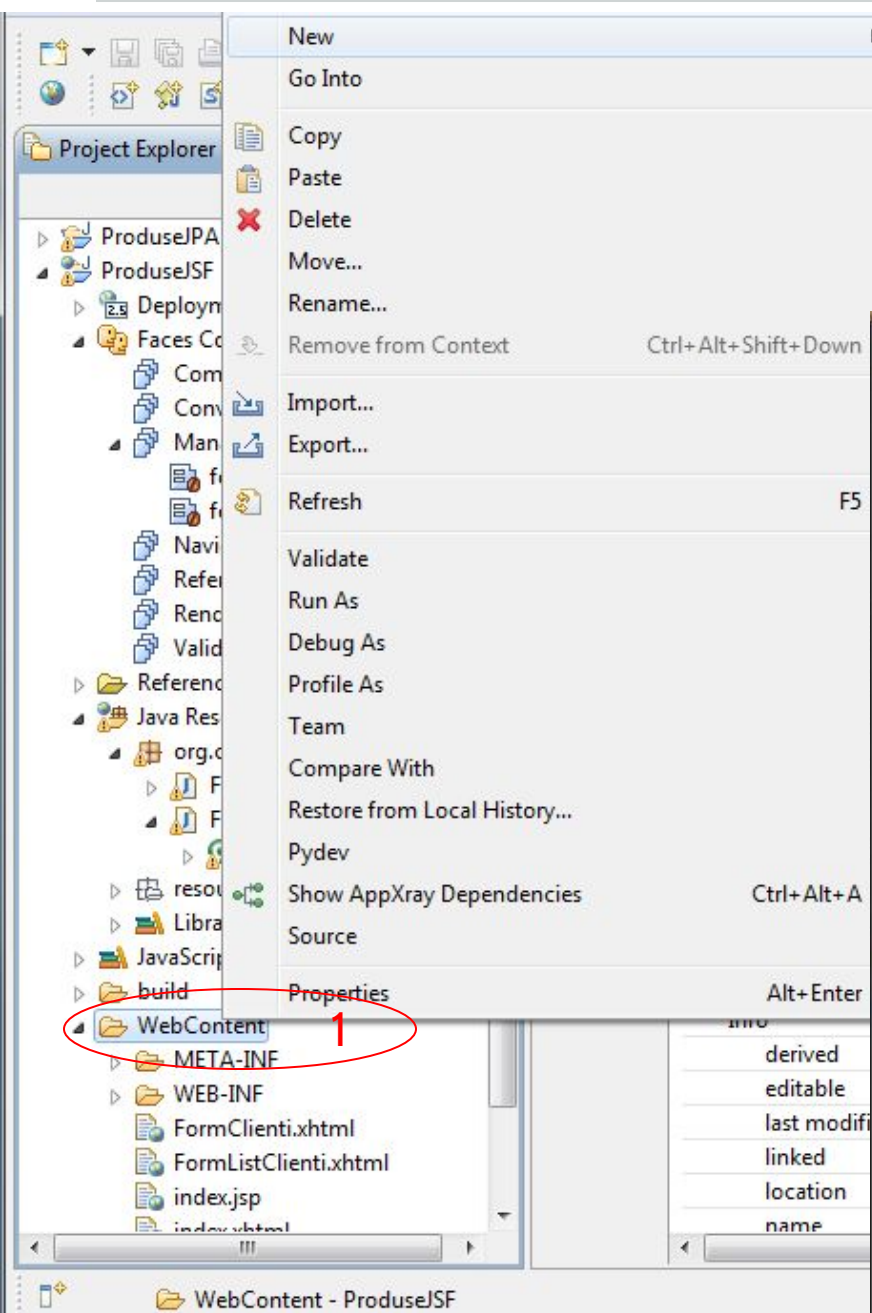


2. Creare formularului principal

- Se urmează întocmai pașii creării unui formular simplu:
 - 2.1 Crearea fișierului *xhtml*;
 - 2.2 Crearea panoului formularului legat la entitatea curentă din *formComenzi*;
 - 2.3 Crearea acțiunilor de navigare-selecție-căutare;
 - 2.4 Crearea acțiunilor tranzacționale.
 - 2.5 Configurări opționale
-

2.1 Crearea fișierului xhtml

- Din secțiunea/folderul *WebContent*
 - Se acționează opțiunea *New HTML file*;
 - Se completează numele *FormComenzi.xhtml*;
 - Se alege șablonul *New Facelet Header*.
- (vezi pe pagina următoare exemplificarea secvenței de pași)



2.2 Crearea panoului formularului principal

- Din secțiunea JSF HTML a paletei de componente :
 - se face drag&drop pentru elementul *Form* pe suprafața formularului după banda cu titlu antetului (headerului);
 - se alege opțiunea de generare: *Generate a form tag and content of data*;
 - se selectează membrul/proprietatea *comanda* a obiectului-suport *formComenzi*;
 - se aranjează rubricile în ordinea dorită și, eventual se modifică etichetele acestora.
- (vezi pe pagina următoare exemplificarea secvenței de pași)

Form comenzi

Palette

- Command Link
- Data Table
- Form**
- Graphic Image
- Head
- Hidden Input
- Link
- Message
- Messages
- Output Format
- Output Label

1

*

Place after div, inside body

New Form

Form Type

Choose the type of form to create.

Generation options: ☐ Configure a form tag

☒ Generate a form tag and content from data

2

Choose Bean/Bean Property

Select a bean or bean property containing fields:

- JSF Managed Beans
 - formClienti
 - formComenzi**
 - comanda**
 - articolle
 - client
 - dataComanda
 - comenzi
 - comenziList
 - produse
 - produseList

3

New Form

Configure Form Fields

Order the fields as they should appear in the form. Indicate the rendering for each field.

Form fields:

Name	Label	Type	Rendering
comanda.id	Id	java.lang.Integ...	Text Field
comanda.data...	Data Comanda	java.util.Date	Text Field
comanda.valoa...	Valoare Coma...	java.lang.Dou...	Text Field

4

< Back

Next >

Finish

Cancel

2.3.1 Crearea acțiunilor de navigare-selecție-căutare (c.)

- (1) Din secțiunea JSF HTML a paletei de componente :
 - (1.1) se face drag&drop pentru elementul *SelectOneMenu* înaintea etichetei primei rubrici a formularului (se forțează o nouă linie);
 - se completează *cboComenzi* pentru *Id*;
 - se completează *#{formComenzi.comanda}* sau selectează (*formComenzi* → *comanda*) pentru *value*;
 - (1.2) se face drag&drop lângă componenta *SelectOneMenu* pentru elementul *PanelGrid* și se completează cu 2 rubrica *columns*;
 - (1.3) se face drag&drop lângă panoul cu 2 coloane pentru un element *Message* cu care se completează noua linie;
- (vezi pe pagina următoare exemplificarea secveței de pași)

Form comenzi

Form Comenzi

Inapoi

Inainte

message

Palette

Command Button 2.1

Command Link 2.2

Data Table

Form

Graphic Image

Head

Hidden Input

Link

Message 1.3

Messages

Output Format

Output Label

Output Link

Output Script

Output Stylesheet

Output Text

Panel Grid 1.2

Panel Group

Secret Input

Select Boolean Checkbox

Select Many Checkbox

Select Many Listbox

Select Many Menu

Select One Listbox

Select One Menu 1.1

New Select One Menu

Select One Menu

Create a new select one menu tag

Select One Menu

Id: cboComenzi 1.1

Value: #{formComenzi.comanda}

Binding:

Validation

Required: ☐

Method:

Finish

Cancel

New Command Button

Command Button

Create a new command button tag

Command Properties

Id: btnPrevious 2.1

Action:

Action listener: #{formComenzi.previousComanda}

Immediate: ☐

Button Properties

Button type: submit

Value: Inapoi

Image:

Finish

Cancel

2.3.2 Crearea acțiunilor de navigare-selecție-căutare (c.)

- (2) Se selectează sub-panoul creat anterior, apoi
 - (2.1) se face drag&drop pentru un element: *Command Button* în cadrul sub-panoului:
 - se completează *btnPrevious* pentru proprietatea *Id*;
 - se completează sau selectează *{formComenzi.previousComanda}* pentru proprietatea *Action Listener*;
 - se completează textul *Inapoi* pentru proprietatea *value*;
 - (2.2) se face drag&drop pentru al doilea element *Command Button* în cadrul sub-panoului:
 - se completează *btnNext* pentru *Id*;
 - se completează sau selectează *{formComenzi.nextComanda}* pentru *Action Listener*;
 - se completează *Inainte* pentru *value*;
- (vezi pe pagina anterioară exemplificarea secveței de pași)

2.3.3 Crearea acțiunilor de navigare-selecție-căutare (completări privind configurarea)

- Se selectează lista combinată, *cboComenzi*, pentru care:
 - (1) se completează proprietatea *converter* cu *#{formComenzi}*;
 - (2) din meniul contextual asociat acestei liste se acționează opțiunea *Insert – Selected Items* și se obține tagul *selectedItems* în cadrul tagului *selectOneMenu* ce marchează controlul grafic al listei inițiale;
 - (3) apoi se completează (sau se selectează în) proprietatea *value* a tagului *selectedItems* cu *#{formComenzi.comenzi}*;
 - (4) din secțiunea *JSF Core* a paletei de componente se aduce prin drag&drop tagul *Ajax* direct în *zona de editare cod* în interiorul tagului principal *selectOneMenu* al listei de selecție-navigare și se completează atributul *render* cu *@form*;
- (vezi în pagina următoare cum ar trebui să arate în final configurația componentei *selectOneMenu* reprezentând lista de navigare-selecție).

Form comenzi

Form Comenzi

Inapoi Inainte

message

```
15 <h:selectOneMenu id="cboComenzi" value="#{formComenzi.comanda}" converter="#{formComenzi}">
16 <p:selectItems value="#{formComenzi.comenzi}"></p:selectItems>
17 <p:ajax render="@form"></p:ajax>
18 </h:selectOneMenu>
19 <h:panelGrid columns="2" border="1">
20 <h:commandButton id="btnPrevious" actionListener="#{formComenzi.previousComanda}" value="Inapoi">
21 <p:ajax render="@form"></p:ajax>
22 </h:commandButton>
23 <h:commandButton id="cmdNext" actionListener="#{formComenzi.nextComanda}" value="Inainte">
24 <p:ajax render="@form"></p:ajax>
25 </h:commandButton>
26 </h:panelGrid>
27
```

Design Preview

Problems Servers Properties Console

General	Property	Value
All	HTML	
	accesskey	
	disabled	
	tabindex	
	JSF	
	binding	
	converter	<code>#{formComenzi}</code>
	id	cboComenzi
	immediate	
	readonly	
	rendered	
	required	
	styleClass	
	validator	
	value	<code>#{formComenzi.comanda}</code>
	valueChangeListener	

2.3.4 Crearea acțiunilor de navigare-selecție-căutare (completări privind configurarea)

- Prin proprietatea *converter* a listei combinate, *cboComenzi*, obiectul *formComenzi* a fost desemnat drept suport pentru:
 - conversia elementelor-valoare ale mapei *<String, Comanda>* *comenzi* din modelul de date, în elemente *String* (vezi operația *getAsString* din clasa *FormComenzi*), astfel că, prin urmare, către formularul HTML va fi transmisă o colecție de asociații *<String, String>*, primul reprezentând eticheta afișată (id-ul comenzii), iar al doilea valoarea elementului din listă (pentru simplificare, tot id-ul comenzii);
 - conversia elementului selectat din formularul HTML pe partea de browser în obiectul *Comanda* corespunzător din modelul de date (vezi operația *getAsObject* din clasa *FormComenzi*);
 - (vezi implementărilor operațiilor *getAsString* și *getAsObject* corespunzătoare în pagina următoare)
-

```
package org.comenzi.forms;
```

```
import java.util.ArrayList;
```

```
public class FormComenzi implements Converter{
```

```
// ... .. //
```

```
// Conversii model pentru interfata grafica
```

```
@Override
```

```
public Object getAsObject(FacesContext arg0, UIComponent uiComponent, String uiValue)
    throws ConverterException {
```

```
    if (uiComponent.getId().equals("cboComenzi")){
        Comanda comandaSablon = new Comanda(Integer.valueOf(uiValue), null);
        return this.comenzi.get(this.comenzi.indexOf(comandaSablon));
    }
```

Implementarea coresp.
listei de comenzi

```
    if (uiComponent.getId().equals("cboProduce")){
        Produs produsSablon = new Produs();
        produsSablon.setCod(Integer.valueOf(uiValue));
        return this.produce.get(this.produce.indexOf(produsSablon));
    }
```

Implementarea coresp.
listei de produse

```
    return null;
```

```
}
```

```
@Override
```

```
public String getAsString(FacesContext arg0, UIComponent uiComponent, Object modelObject)
    throws ConverterException {
```

```
    if (uiComponent.getId().equals("cboComenzi")){
        return ((Comanda)modelObject).getId().toString();
    }
```

Implementarea coresp.
listei de comenzi

```
    if (uiComponent.getId().equals("cboProduce")){
        return ((Produs)modelObject).getCod().toString();
    }
```

Implementarea coresp.
listei de produse

```
    return null;
```

```
}
```

2.4 Crearea acțiunilor formularului principal

- Din secțiunea JSF HTML a paletei de componente:
 - se face drag&drop pentru 4 componente *commandButton* chiar sub panoul principal al formularului master;
 - fiecare buton de comandă
 - va avea un *Id* distinct: *cmdAdaugareComanda*, *cmdStergereComanda*, *cmdSalvare*, *cmdAbandon*;
 - va afișa un text distinct prin proprietatea *value*: Adaugare Comanda, Stergere Comanda, Salvare, Abandon;
 - va fi legat la o operație tranzacțională specifică a *formComenzi* prin proprietatea *actionListener*:
 - {formComenzi.adaugareComanda},
 - {formComenzi.stergeComanda},
 - {formComenzi.salvareComanda},
 - {formComenzi.abandon}
- (vezi pe pagina următoare dispunerea celor patru butoane (1), proprietățile butonului *cmdAbandon*(2) și implementarea acțiunilor tranzacționale (3) în clasa *FormComenzi*)

Form Comenzi

<input type="text"/>	Inapoi Inainte	message
Id:	<input type="text" value="#{formComenzi.comanda}"/>	message
Data Comanda:	<input type="text" value="#{formComenzi.comanda}"/>	message
Valoare Comanda:	<input type="text" value="#{formComenzi.comanda}"/>	message
<input type="button" value="Adaugare comanda"/> <input type="button" value="Stergere comanda"/> <input type="button" value="Salvare"/> <input type="button" value="Abandon"/>		

```

42 <h:commandButton id="cmdAdaugareComanda" value="Adaugare comanda"
43     actionListener="#{formComenzi.adaugareComanda}"></h:commandBut
44 <h:commandButton id="cmdStergereComanda" value="Stergere comanda"
45     actionListener="#{formComenzi.stergereComanda}"></h:commandBut
46 <h:commandButton id="cmdSalvare" value="Salvare"
47     actionListener="#{formComenzi.salvareComanda}"></h:commandButt
48 <h:commandButton id="cmdAbandon" value="Abandon"
49     actionListener="#{formComenzi.abandon}"></h:commandButton>
50

```

Design Preview

Problems Servers Properties Console

General

Command Properties

Id: cmdAbandon

Action:

Action listener: #{formComenzi.abandon}

Immediate: ☐

Button Properties

Button type: submit

Value: Abandon

FormComenzi.java FormComenzi.xhtml

```

21
22 public class FormComenzi implements Converter{
23     // ... .. //
24     // Actiuni tranzactionale
25     public void adaugareComanda(ActionEvent evt){
26         Comanda comandaNoua = new Comanda(999, new Date());
27         this.comenzi.add(comandaNoua);
28         this.comanda = comandaNoua;
29     }
30
31     public void stergereComanda(ActionEvent evt){
32         if (this.em.contains(this.comanda)){
33             this.em.getTransaction().begin();
34             this.em.remove(this.comanda);
35             this.em.getTransaction().commit();
36             this.comenzi.remove(this.comanda);
37         }
38         if (this.comenzi.size() > 0)
39             this.comanda = this.comenzi.get(0);
40         else
41             this.comanda = null;
42     }
43
44     public void salvareComanda(ActionEvent evt){
45         this.em.getTransaction().begin();
46         this.em.persist(this.comanda);
47         this.em.getTransaction().commit();
48     }
49
50     public void abandon(ActionEvent evt){
51         if (this.em.contains(this.comanda)){
52             this.em.getTransaction().begin();
53             this.em.refresh(this.comanda);
54             this.em.getTransaction().commit();
55         }else{
56             if (this.comenzi.size() > 0)
57                 this.comanda = this.comenzi.get(0);
58         }
59     }

```

2.5 Configurări opționale ale formularului principal

- 1. Adăugarea unui format corespunzător pentru data calendaristică
 - din meniul contextual (click dr.) asociat componentei *inputText* care prezintă **data comenzii** se alege opțiunea *Insert – Convertor – Date Time*, obținându-se astfel tagul *converterDateTime*;
 - în fereastra de proprietăți ale tagului *converterDateTime*, se completează rubrica *pattern* din pagina *All* cu formatul-șir de caractere: *dd/MM/yyyy*
- 2. Expunerea proprietății calculate *valoareComanda* printr-o componentă read-only
 - se poate realiza prin
 - modificarea proprietății *readonly=false* a actualei componente de tip *inputText*;
 - Sau ștergerea componentei inițiale de tip *inputText*, urmată de înlocuirea acesteia cu o nouă componentă de tip *outputText* din secțiunea JSF HTML a paletei de componente care va avea proprietatea *value* legată la proprietatea *valoareComanda* a comenzii curente din modelul de date prin expresia `{formComenzi.comanda.valoareComanda}`
- (vezi pe pagina următoare proprietățile convertorului pentru dată calendaristică și ale componentei read-only de tip *outputText*).

Form Comenzi

Inapoi

Inainte

message

Id:

#{formComenzi.comanda

message

Data Comanda:

#{formComenzi.comanda

message

Valoare Comanda:

#{formComenzi.comanda

message

Adaugare comanda

Stergere comanda

Salvare

Abandon

```

33      <h:outputText value="Data Comanda:"></h:outputText>
34      <h:inputText id="dataComanda"
35          value="#{formComenzi.comanda.dataComanda}"
36          <p:convertDateTime pattern="dd/MM/yyyy"></p:convertDateTime>
37      </h:inputText>
38      <h:message for="dataComanda"></h:message>
39      <h:outputText value="Valoare Comanda:"></h:outputText>
40      <h:inputText id="valoareComanda"
41          value="#{formComenzi.comanda.valoareComanda}"></h:inputText>

```

Convertorul pentru
data calendarică

Design Preview

Problems Servers Properties Console

General

All

Property	Value
Attributes	
binding	
dateStyle	
for	
locale	
pattern	dd/MM/yyyy

3. Construirea sub formularului detail

- 3.1 Adăugarea sub-formularului folosind un tabel-grid
 - 3.2 Configurarea unei coloane cu listă de selecție
 - 3.3 Configurarea acțiunilor specifice sunformuarului
 - adăugare acțiune ștergere linii din grd;
 - adăugare linii-detalii în grid.
-

3.1 Adăugarea sub-formularului folosind un tabel-grid

- Cea mai rapidă modalitate de a obține un tabel-grid legat la o colecție de date constă în:
 - (1) localizarea colecției în secțiunea *Data Palette* a paletii de componente: în cazul nostru localizarea obiectului suport *formComenzi* în categoria *JSF Managed Beans*, obiect care conține proprietatea *comanda* corespunzătoare comenzii curente prezentată prin formularul master, iar obiectul-valoare a proprietății *comanda* conține colecția de *articole* pe baza căreia vom crea subformularul detail;
 - (2) aducerea prin drag&drop a colecției localizate în paleta de componente undeva sub (sau în dreapta) formularul principal și butoanele de acțiune ale acestuia;
 - configurarea tabelului grid prin paginile asistentului care va fi activat prin acțiunea drag&drop:
 - (3.1) alegerea tipului de generator grafic: JSF Data Table;
 - (3.2) stabilirea numelui variabilei interne a tabelului, variabilă internă care va desemna (prin interacție) fiecare element din colecția sursă de date;
 - (3.3) stabilirea ordinii coloanelor din tabelul grid.
-

Form comenzi

The screenshot displays a web application development environment. The main window shows a form titled "Form Comenzi" with fields for "Id", "Data Comanda", and "Valoare Comanda". Two "Insert Enumeration" dialog boxes are open, illustrating the configuration of the form's data table.

3.1 (in the first dialog): The "Choose Generator" tab is selected, and "JSF Data Table" is chosen as the generator.

3.2 (in the second dialog): The "Iteration Variable" tab is selected. The "Name" field is set to "linieComanda".

3.3 (in the second dialog): The "Order Fields" tab is selected. The fields are ordered as follows:

- id
- produs
- cantitate
- valoareArticol

The "Palette" on the right shows the "Data Palette" with the "comanda" bean selected. The "articole" property is highlighted, indicating the data source for the iteration.

3. 1 Adăugarea sub-formularului folosind un tabel-grid (c)

- **Definiția fiecărei coloane** (tagul column) a tabelului-grid presupune:
 - un antet desemnat printr-un subtag de tip *facet* numit *header*;
 - o componentă care va fi multiplicată pentru fiecare linie, componentă
 - legată la sursa de date prin intermediul proprietății *value*;
 - a cărei tip implicit este `outputText` – prin urmare pentru coloanele editabile trebuie înlocuită o altă componentă de tip corespunzător (de ex. `inputText` – vezi modificările pe coloanele *cantitate* și *produs*);
 - sursa de date pentru coloane este desemnată prin proprietăți ale variabilei (*linieComanda*, în cazul nostru) desemnate să parcurge (itereze) elementele coloanei sursă de date a gridului.
- (Vezi modul de configurare a coloanei *cantitate* în pagina următoare)
- Este posibil ca noua componentă *dataTable* reprezentând gridul cu detaliile formularului să nu aibă stabilit un “stil” predefinit: minimal modificați proprietatea *Border Size* (din pagina Table a ferestrei Properties) la valoare 1.

Form Comenzi

Inapoi

Inainte

message

Id:

#{formComenzi.comanda}

message

Data Comanda:

#{formComenzi.comanda}

message

Valoare Comanda:

#{formComenzi.comanda}

message

Adaugare comanda

Stergere comanda

Salvare

Abandon

Id

#{linieComanda.id}

Produs

Cantitate

#{linieComanda.cantitate}

Valoare Articol

#{linieComanda.valoareArticol}

```
69</h:column>
70<h:inputText id="txtCantitate" value="#{linieComanda.cantitate}"></h:inputText>
71<p:facet name="header">
72  <h:outputText value="Cantitate"></h:outputText>
73</p:facet>
74
75</h:column>
```

DesignPreview

ProblemsServersPropertiesConsole

General

Input Text txtCantitate

All

Id:txtCantitate

Value:#{linieComanda.cantitate}

Palette

Output Script

Output Stylesheet

Output Text

Panel Grid

Panel Group

Secret Input

Select Boolean Checkbox

Select Many Checkbox

Select Many Listbox

Select Many Menu

Select One Listbox

Select One Menu

Select One Radio

Text Input

Textarea Input

JSTL Core

New Input Text

Input Text

Create a new input text tag

Input Text

Id:txtCantitate

Value:#{linieComanda.cantitate}

Binding:

Size:

Converter:

Validation

Required:

Method:

Finish

Cancel

3.2 Configurarea unei coloane cu listă de selecție

- Coloana corespunzătoare produsului fiecărui articol al comenzii curente a fost configurată implicit folosind componenta *inputText*, componentă puțin corespunzătoare, ținând seama că valorile editabile din această coloană se referă la instanțe *Produs*;
- Componenta din coloana **Produs** va trebui **înlocuită** cu o listă de selecție astfel:
 - se renunță la (se șterge) componenta *outputText* inițială;
 - (1) se aduce în locul acesteia o componentă ***selectOneMenu***:
 - (2) legată la proprietatea *produs* a articolului accesibil prin variabila *linieComanda* prin rubrica *value*;
 - (3) legată la obiectul suport *formComenzi*, prin proprietatea *converter*, pentru conversia produselor în valorile Integer ale *id-urilor acestora* (vezi suportul de conversie discutat anterior pentru rubrica *cboComenzi*);
 - (4) alimentată prin sub-tagul *selectedItems* din colecția *produse* accesibilă prin proprietatea corespunzătoare a obiectului-suport *formComenzi*.
 - (vezi în pagina următoare configurarea listei de produse)

This image shows a screenshot of the Eclipse IDE with the JSP Editor, Design View, and several configuration dialogs for a JSP page.

Design View: The top part of the image shows the Design View of a JSP page. It contains a table with columns: Id, Data Comanda, and Valoare Comanda. Below this is a table with columns: Id, Produs, Cantitate, and Valoare Articol. The **Produs** column is highlighted with a red circle and the number 1.

Code View: The bottom part of the image shows the Code View of the same JSP page. The code is as follows:

```
<h:column>
  <h:selectOneMenu id="cboProdus" value="#{linieComanda.produs}" converter="#{formComenzi}">
    <p:selectItems value="#{formComenzi.produs}">
    </p:selectItems>
  </h:selectOneMenu>
  <p:facet name="header">
    <h:outputText value="Produs"></h:outputText>
  </p:facet>
</h:column>
```

The **converter="#{formComenzi}"** attribute is circled in red with the number 4.

Properties View: The bottom left shows the Properties View. The **converter** property is highlighted with a red circle and the number 3.

New Select One Menu Dialog: The middle right dialog is titled "New Select One Menu". It contains the following fields:

- Id:** cboProdus
- Value:** #{linieComanda.produs}
- Binding:** (empty)

The **Value** field is circled in red with the number 2.

Choose Binding Dialog: The bottom right dialog is titled "Choose Binding". It contains a tree view of the page variables and JSF managed beans. The **produs** property of the **linieComanda** bean is selected and circled in red with the number 2.

3. 3 Acțiuni specifice sub-formularului detail

- Acțiunea de *ștergere* implică asocierea fiecărei linii cu un buton de comandă corepunzător, lucru care nu implică un mecanism special pentru a desemna linia curent selectată;
 - Acțiunea de *adăugare* va fi reprezentată de un singur buton de comandă care poate fi plasat sub tabelul-grid cu detalii-articole
-

3.3.1 Configurarea unei coloane pentru acțiunea de ștergere

- Pentru a obține butoanele corespunzătoare acțiunii de ștergere:
 - (1) va trebui creată o coloană specifică astfel: se selectează ultima coloană curentă (*valoare*), apoi din meniul contextual (click dr.) va fi acționată opțiunea *Data Table – Insert Column After*;
 - noua coloană astfel creată:
 - va avea un antet corespunzător cu textul *Șterge*;
 - (2) va fi completată cu componenta *command Button* din paleta de componente, componentă care va fi configurată astfel:
 - *Id*: *cmdStergeArticole*;
 - *Value*: *Șterge*;
 - *ActionListener*: *#{formComenzi.stergeArticol}*
- (vezi în următoarele două pagini crearea coloanei și configurarea butoanelor de ștergere).

FormComenzi.java FormComenzi.xhtml

Id:	<input type="text" value="#{formComenzi.comanda}"/>	message
Data Comanda:	<input type="text" value="#{formComenzi.comanda}"/>	message
Valoare Comanda:	<input type="text" value="#{formComenzi.comanda.valoareComanda}"/>	message

Adaugare comanda Stergere comanda Salvare Abandon

Id	Produs	Cantitate	Valoare Articol
<input type="text" value="#{linieComanda.id}"/>	<input type="text" value="#{linieComanda.produs}"/>	<input type="text" value="#{linieComanda.cantitate}"/>	<input type="text" value="#{linieComanda.valoare}"/>

1

Palette

- HTML 4.0
- JSF Core
- JSF HTML
 - Body
 - Button
 - Column
 - Command Button
 - Command Link
 - Data Table
 - Form

Edit
Select
Insert
DataTable
Show

Select Table
Insert Column Before
Insert Column After
Delete Column
Insert Table Header
Insert Table Footer
Delete Table Header
Delete Table Footer
Insert Column Header
Insert Column Footer
Delete Column Header
Delete Column Footer

```
77 </h:column>
78 <h:column>
79   <p:facet name="header">
80     <h:outputText value="Valoare Articol"></h:outputText>
81   </p:facet>
82   <h:outputText id="valoareArticol"
83     value="#{linieComanda.valoareArticol}"></h:outputText>
84 </h:column>
85 </h:dataTable>
86 </h:form>
```

Design Preview

Problems Servers Properties Console

Tomcat v6.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre6\bin\javaw.exe (Dec 18, 2010 4:07:41 PM)

Form Comenzi

<input type="text"/>	<input type="button" value="Inapoi"/> <input type="button" value="Inainte"/>	message
Id:	<input type="text" value="#{formComenzi.comanda}"/>	message
Data Comanda:	<input type="text" value="#{formComenzi.comanda}"/>	message
Valoare Comanda:	<input type="text" value="#{formComenzi.comanda.valoareComanda}"/>	message

Id	Produs	Cantitate	Valoare Articol	Column
<input type="text" value="#{linieComanda.id}"/>	<input type="text"/>	<input type="text" value="#{linieComanda.cantitate}"/>	<input type="text" value="#{linieComanda.valoareArticol}"/>	<input type="button" value="Sterge"/>

```

84      </h:column>
85      <h:column>
86          <h:commandButton id="cmdStergeArticol"
87              actionListener="#{formComenzi.stergeArticol}" value="Sterge" />
88      </h:column>
89      <p:facet name="header">
90          <h:outputText value="Column"></h:outputText>
91      </p:facet>

```

Design Preview

Problems Servers Properties Console

General

Command Properties

Id: cmdStergeArticol
Action:
Action listener: #{formComenzi.stergeArticol}
Immediate: ☐

Button Properties

Button type: submit
☒ **Value:** Sterge

Palette
 HTML 4.0
 JSF Core
 JSF HTML
 Body
 Button
 Column
 Command Button
 Command Link
 Data Table
 Form
 Graphic Image
 Head
 Hidden Input
 Link
 Message

New Command Button

Create a new command button tag

Command Properties

Id: cmdStergeArticol
Action:
Action listener: #{formComenzi.stergeArticol}
Immediate: ☐

Button Properties

Button type: submit
☒ **Value:** Sterge
☐ **Image:**

3.3.1 Configurarea unei coloane pentru acțiunea de ștergere (c.)

- Butonul de comandă din noua coloană va fi legat la acțiunea de ștergere, însă va trebui să transmită și o informație cu privire la identificarea liniei-detaliu care va constitui subiectul acțiunii de ștergere. Acest lucru poate fi realizat cu ajutorul unui atribut specific asociat cu fiecare instanță a butonului de ștergere și a cărui valoare va fi legată la proprietatea *id* a elementelor din sursa de date accesibilă prin variabila de iterație *linieComandă*.
- Un astfel de atribut va fi obținut astfel:
 - (1) din meniul contextual (click dr.) asociat butonului de comandă va fi accesată opțiunea *Insert – Attribute*;
 - în fereastra de configurare corespunzătoare noului atribut:
 - (2.1) va fi specificat numele atributului: *name – selectedId*;
 - (2.2) va fi legată proprietatea *value* la *#{linieComanda.id}*.
- (vezi în următoarea pagină crearea și configurarea atributului suplimentar specific butoanelor de ștergere)

Form Comenzi

Id: #{formComenzi.comanda} message

Data Comanda: #{formComenzi.comanda} message

Valoare Comanda: #{formComenzi.comanda.valoareComanda} message

Adaugare comanda Stergere comanda Salvare Abandon

Id	Produs	Cantitate	Valoare Articol	Column
#{linieComanda.id}		#{linieComanda.cantitate}	#{linieComanda.valoareArticol}	

85< <h:column>
86< <h:commandButton id="cmdStergeArticol"
87actionListener="#{formComenzi.stergeArticol}" value="St
88<p:attribute name="selectedId" value="#{linieComanda.id
89</p:attribute>
90</h:commandButton>
91<p:facet name="header">
92<h:outputText value="Column"></h
93</p:facet>
94</h:column>
95</h:dataTable>

Design Preview

Problems Servers Properties Console

General Command Properties

All

Id: cmdStergeArticol

Action:

Action listener: #{formComenzi.stergeArticol}

Immediate:

Palette

- HTML 4.0
- JSF Core
- JSF HTML
 - Body
 - Button
 - Column
 - Command Button
 - Command Link
 - Data Table
 - Form
 - Graphic Image
 - Head
 - Hidden Input
 - Link

Edit Select Insert DataTable Show

message messages ActionListener Attribute Converter Param

1

Create attribute

Create subelement "attribute" under "h:commandButton"

name: selectedId 2.1

value: #{linieComanda.id} 2.2

OK Cancel

În clasa *FormComenzi* va fi creată operația *stergeArticol* care presupune:

- citirea valorii atributului *selectedId*;
- crearea unui obiect-șablon-de-căutare pe baza valorii id-ului selectat prin care va fi desemnat elementul ce va fi șters din colecția de articole a comenzii curente.

Acțiunea de crearea a unei linii-detaliu (*adaugareArticol*) presupune, în cazul de față, crearea unei instanțe *ArticolComanda* și adăugarea acesteia în modelul de date – colecția de articole a comenzii corente, colecția care formează sursa de date a tabelului grid.



```
1 package org.comenzi.forms;
2
3 import java.util.ArrayList;
4
21
22 public class FormComenzi implements Converter{
23     // ... .. //
24     public void adaugaArticol(ActionEvent evt){
25         ArticolComanda articolNou = new ArticolComanda(9999, this.produce.get(0), 0.0);
26         this.comanda.getArticole().add(articolNou);
27         articolNou.setComanda(this.comanda);
28     }
29
30     public void stergeArticol(ActionEvent evt){
31         Integer selectedId = Integer.valueOf(evt.getComponent().getAttributes().get("selectedId").toString());
32         ArticolComanda articolSablon = new ArticolComanda();
33         articolSablon.setId(selectedId);
34         this.comanda.getArticole().remove(articolSablon);
35     }
36 }
```

3.3.1 Configurarea acțiunii de adăugare

- Din paleta de componente se va aduce un nou buton de comandă (1) care va fi legat la operațiunea *adaugareArticol* din clasa *FormComenzi* (2).
 - (vezi în următoarea pagină crearea și configurarea butonului de adăugare).
-

New Command Button

Create a new command button tag

Command Properties

Id: cmdAaugareArticol 2.1

Action:

Action listener: #{formComenzi.adaugaArticol} 2.2

Immediate: ☐

Button Properties

Button type: submit

Value: Aauga articol 2.3

Image:

Finish Cancel

Form Comenzi

Id:

Data Comanda:

Valoare Comanda:

Adauga comanda Sterge

Id:

Produs:

Car:

Adauga articol

Palette

- HTML 4.0
- JSF Core
- JSF HTML
 - Body
 - Button
 - Column
 - Command Button 1**
 - Command Link
 - Data Table
 - Form
 - Graphic Image
 - Head
 - Hidden Input
 - Link
 - Message
 - Messages
 - Output Format

```
92 <h:outputText value="Sterge"></h:outputText>
93 </p:facet>
94 </h:column>
95 </h:dataTable>
96 <h:commandButton id="cmdAaugareArticol"
97   actionListener="#{formComenzi.adaugaArticol}" value="Aauga articol">
98 </h:commandButton>
99 </h:form>
100 </body>
```

În final, rezultatul ar trebui să fie următorul:

Mozilla Firefox

http://localhost:8081/ProduseJSF/FormComenzi.jsf

Most Visited Getting Started Latest Headlines NetBeans IDE Download PortalFEAA

http://localhost:8...SF/FormComenzi.jsf

Form comenzi

Form Comenzi				
1001 ▾	<input type="button" value="Inapoi"/> <input type="button" value="Inainte"/>			
Id:	1001			
Data Comanda:	17/12/2010			
Valoare Comanda:	100.0			
<input type="button" value="Adaugare comanda"/> <input type="button" value="Stergere comanda"/> <input type="button" value="Salvare"/> <input type="button" value="Abandon"/>				
Id	Produs	Cantitate	Valoare Articol	Sterge
2	Produs 101 ▾	5.0	50.0	<input type="button" value="Sterge"/>
1	Produs 101 ▾	5.0	50.0	<input type="button" value="Sterge"/>
<input type="button" value="Adauga articol"/>				

Observații

- Generarea coloanelor în grid-dataTable se face implicit cu componente *outputText* care tb convertite în *inputText* acolo unde este nevoie.
 - Dacă nu sunt date pentru a popula modelul de inițial, trebuie adăugată o entitate selectabilă (nouă) implicită.
-