

---

# L8. TUTORIAL Eclipse [II]

## Java Server Faces: Form Simplu Complet

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

---

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

---

# Plan

---

- 1. Implementare acțiuni CRUD
    - Codificare acțiuni
    - Invocare acțiuni prin butoane de comandă
  - 2. Navigare prin selectare în listă-combinată a entității curente
    - adăugare componentă de selecție-navigare
    - legare rubrică selecție-navigare la model de date
    - codificare operații de conversie-sincronizare
  - 3. Test formular start JSF
-

# 1. Implementare acțiuni CRUD

---

- 1.1 Codificare acțiuni CRUD
  - 1.2 Adăugare butoane de comandă (de acțiune) în formularul grafic
  - 1.3 Legare butoane de comandă la acțiuni
-

## Codul sursă al controller-beanului curent al formularului

```
FormClienti.java X FormClienti.xhtml

public class FormClienti {
    private Client client;
    private List<Client> clienti =
        new ArrayList<Client>();
    public Client getClient() {
        return client;
    }
    public void setClient(Client client) {
        this.client = client;
    }
    public List<Client> getClienti() {
        return clienti;
    }
    public void setClienti(List<Client> clienti) {
        this.clienti = clienti;
    }

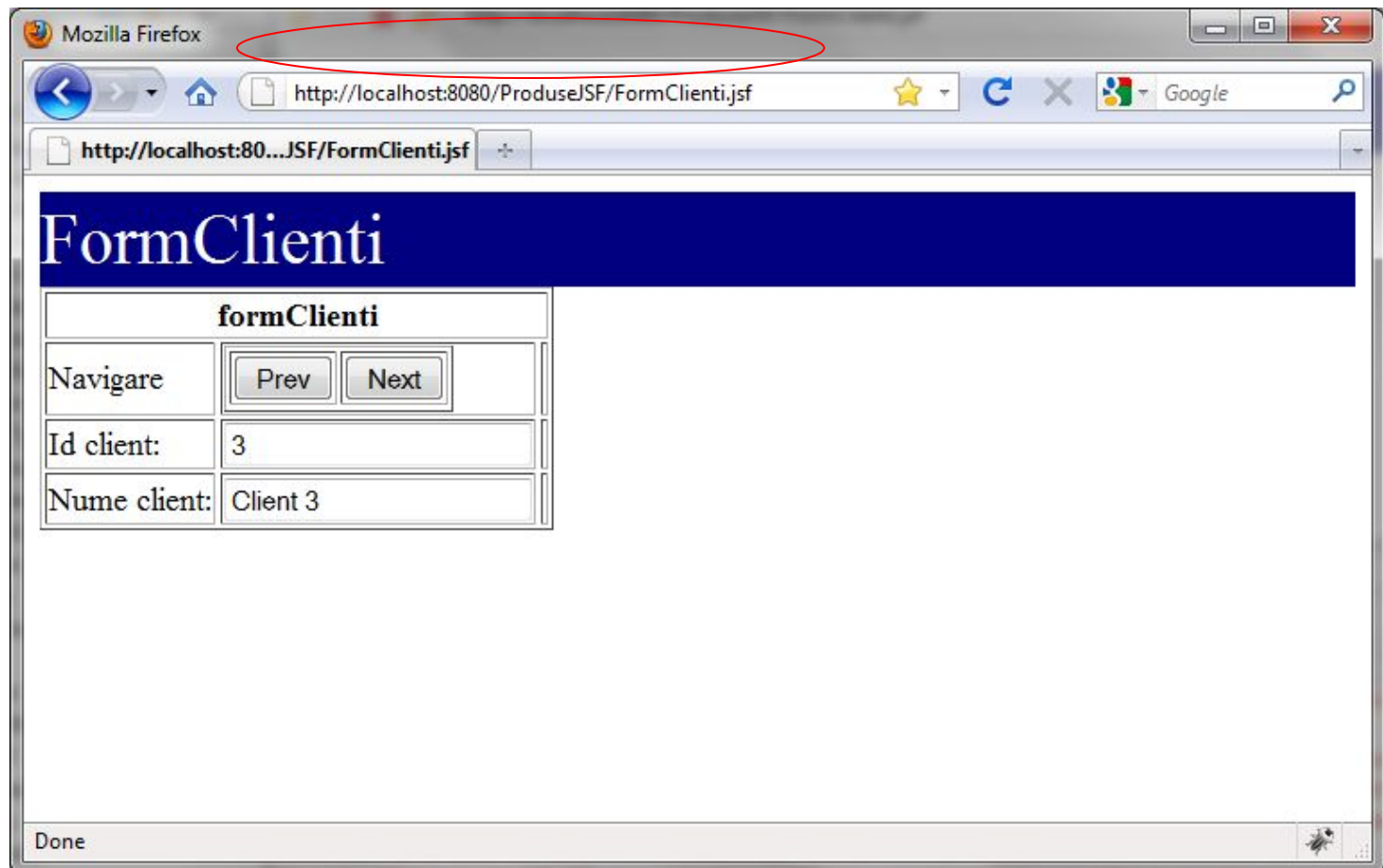
    private EntityManager em;
    public FormClienti() {
        EntityManagerFactory emf =
            Persistence.createEntityManagerFactory("ProduseJPA");
        em = emf.createEntityManager();
        this.clienti = em.createQuery("SELECT c FROM Client c")
            .getResultList();
        if (!this.clienti.isEmpty())
            client = this.clienti.get(0);
    }

    public void previousClient(ActionEvent evt){
        Integer idxCurent = this.clienti.indexOf(client);
        if (idxCurent > 0)
            this.client = this.clienti.get(idxCurent - 1);
    }

    public void nextClient(ActionEvent evt){
        Integer idxCurent = this.clienti.indexOf(client);
        if ((idxCurent+1) < this.clienti.size())
            this.client = this.clienti.get(idxCurent + 1);
    }
}
```

# Starea curentă a formularului

---



Mozilla Firefox

http://localhost:8080/ProduseJSF/FormClienti.jsf

http://localhost:80...JSF/FormClienti.jsf

## FormClienti

formClienti	
Navigare	<input type="button" value="Prev"/> <input type="button" value="Next"/>
Id client:	3
Nume client:	Client 3

Done

# 1.1. Implementare acțiuni CRUD

---

- Acțiunile **CRUD** (adăugare, ștergere, abandon, salvare) vor fi codificate în fișierul bean-controller `FormClienti.java`:
    - *public void **adaugareClient**(ActionEvent evt):* se creează un nou client și se adăugă în modelul de date;
    - *public void **stergeClient**(ActionEvent evt):* se îndepărtează clientul curent din modelul de date și din contextul de persistență (*EntityManager*);
-

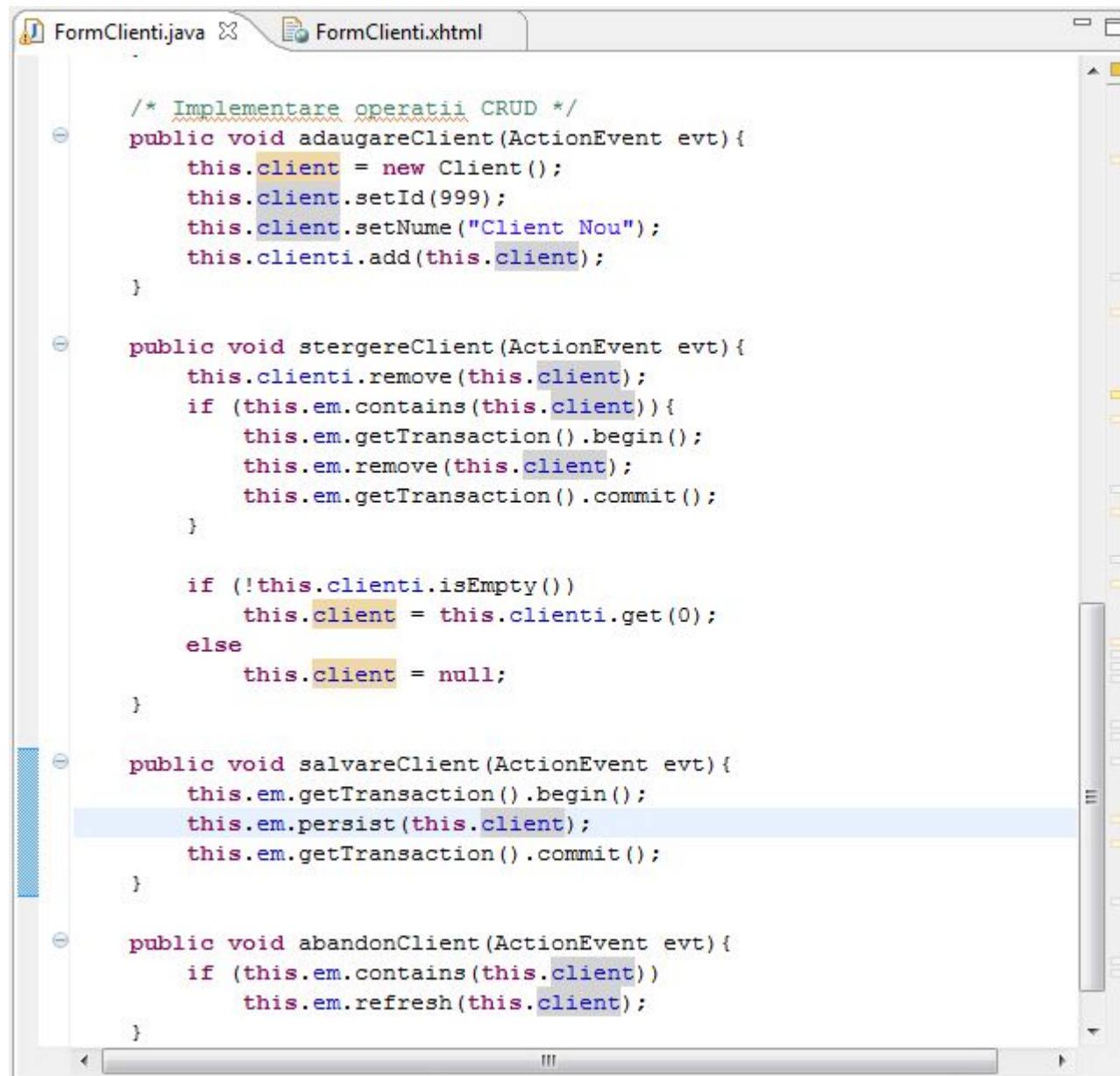
# Implementare acțiuni CRUD

---

- Acțiunile CRUD din FormCienti.java:
  - *public void **abandonClient**(ActionEvent evt):* se resincronizează clientul curent cu starea actuală din suportul de persistență (schema BD la care este conenctat *entityManagerul* formularului);
  - *public void **salveazaClient**(ActionEvent evt):* se asociază clientul curent cu contextul de persistență (*EntityManager*) și se declanșează tranzacția cu suportul de persistență (schema BD).

***Vezi codul sursă în pagina următoare !!!***

---



The screenshot shows an IDE window with two tabs: 'FormClienti.java' and 'FormClienti.xhtml'. The 'FormClienti.java' tab is active, displaying Java code for CRUD operations. The code is as follows:

```
/* Implementare operatii CRUD */
public void adaugareClient(ActionEvent evt){
    this.client = new Client();
    this.client.setId(999);
    this.client.setName("Client Nou");
    this.clienti.add(this.client);
}

public void stergereClient(ActionEvent evt){
    this.clienti.remove(this.client);
    if (this.em.contains(this.client)){
        this.em.getTransaction().begin();
        this.em.remove(this.client);
        this.em.getTransaction().commit();
    }

    if (!this.clienti.isEmpty())
        this.client = this.clienti.get(0);
    else
        this.client = null;
}

public void salvareClient(ActionEvent evt){
    this.em.getTransaction().begin();
    this.em.persist(this.client);
    this.em.getTransaction().commit();
}

public void abandonClient(ActionEvent evt){
    if (this.em.contains(this.client))
        this.em.refresh(this.client);
}
```



## 1.2 Adăugare butoane de comandă în formular

---

- Se va obține o nouă linie în formular prin aducerea a două componente grafice *PanelGrid* din secțiunea *JSF HTML* a paletei de componente, fiecare nou panou având 2 coloane. Linia nouă este completată eventual cu un *Message* sau *OutputText* (reamintim că panoul principal al formularului are trei coloane);

FormClienti.java \*FormClienti.xhtml

# Formular Clienti

Formular Clienti

Navigare	<div>Prev</div> <div>Next</div>	message
Id:	<code>{formClienti.client.id}</code>	message
Nume:	<code>{formClienti.client.nume}</code>	message
		message

Palette

- Form
- Graphic Image
- Head
- Hidden Input
- Link
- Message
- Messages
- Output Format
- Output Label
- Output Link
- Output Script
- Output Stylesheet
- Output Text
- Panel Grid

New Panel Grid

Panel Grid Type

Choose the type of panel grid to create.

Generation options: ☒ Configure a panelGrid tag ☐ Generate a panelGrid tag and content from data

Table Properties

Columns:

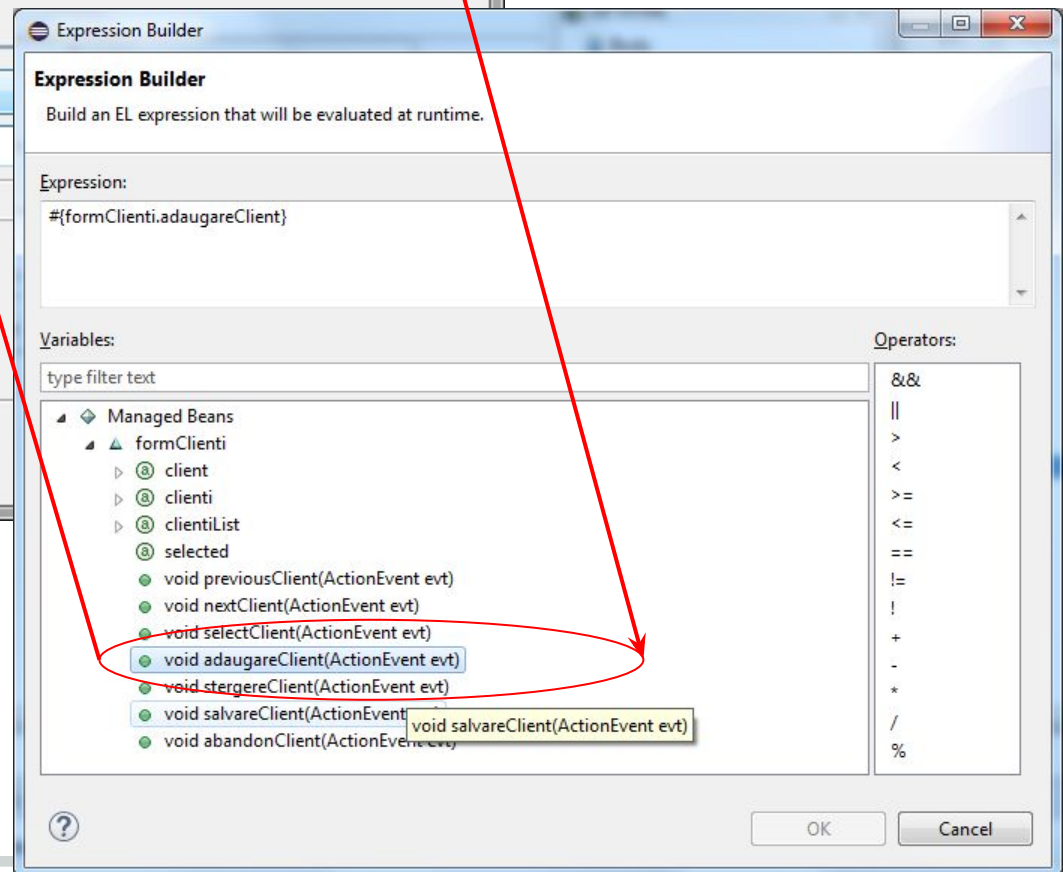
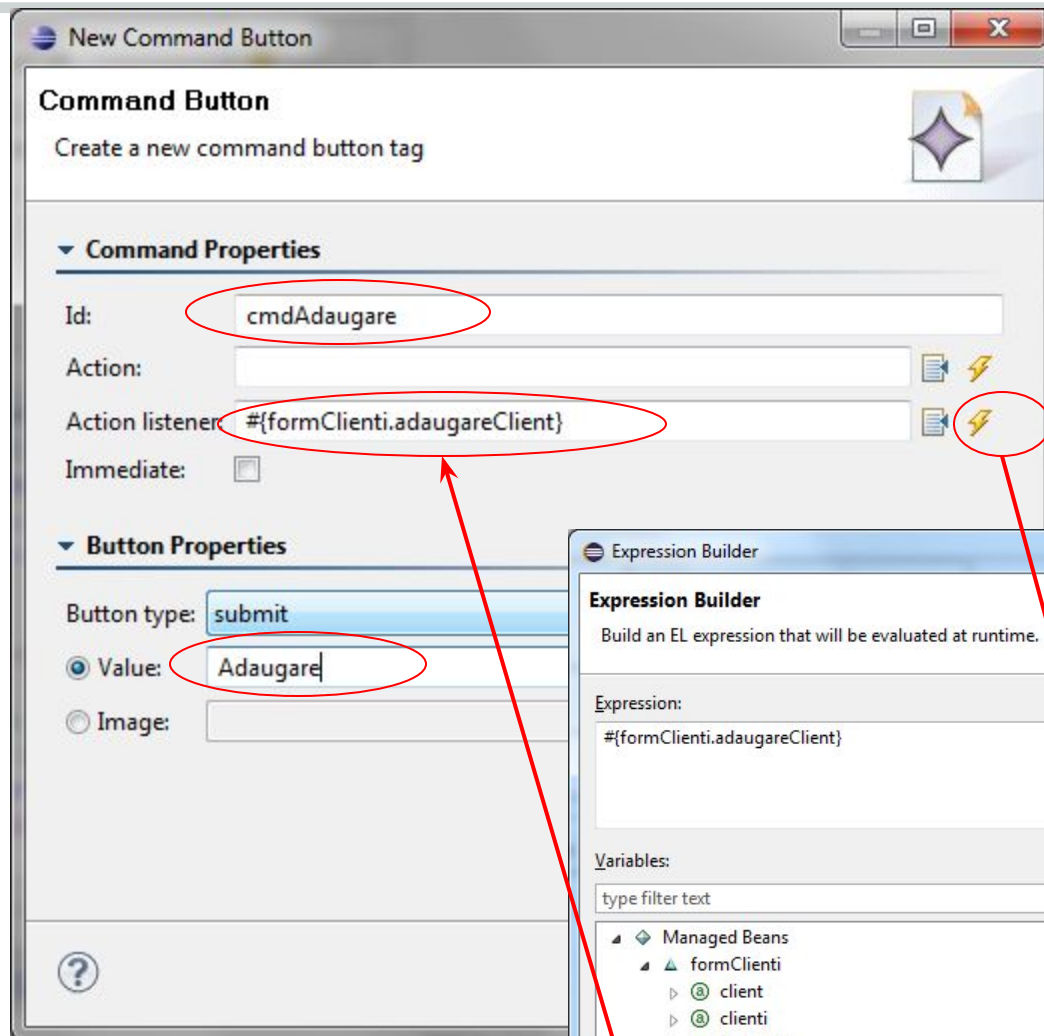
Width:

Table class:

# 1.2 Adăugare butoane de comandă în formular

---

- În fiecare sub-panou de pe ultima linie se vor adăuga câte 2 butoane de comandă astfel:
  - *Primul panou:*
    - buton pentru adăugare:
      - *Id:* **cmdAdaugare**
      - *Value:* **Adaugare** (textul afișat pe buton)
      - *Action Listener:* se va acționa butonul **Bind to a dynamic value** din fereastra de proprietăți și se va selecta: *formClienti* - *adaugareClient*, rezultând expresia ***#{formClienti.adaugareClient}*** (Vezi pagina următoare!)
    - buton pentru stergere:
      - *Id:* **cmdStergere**
      - *Value:* **Stergere**
      - *Action Listener:* ***#{formClienti.stergereClient}***



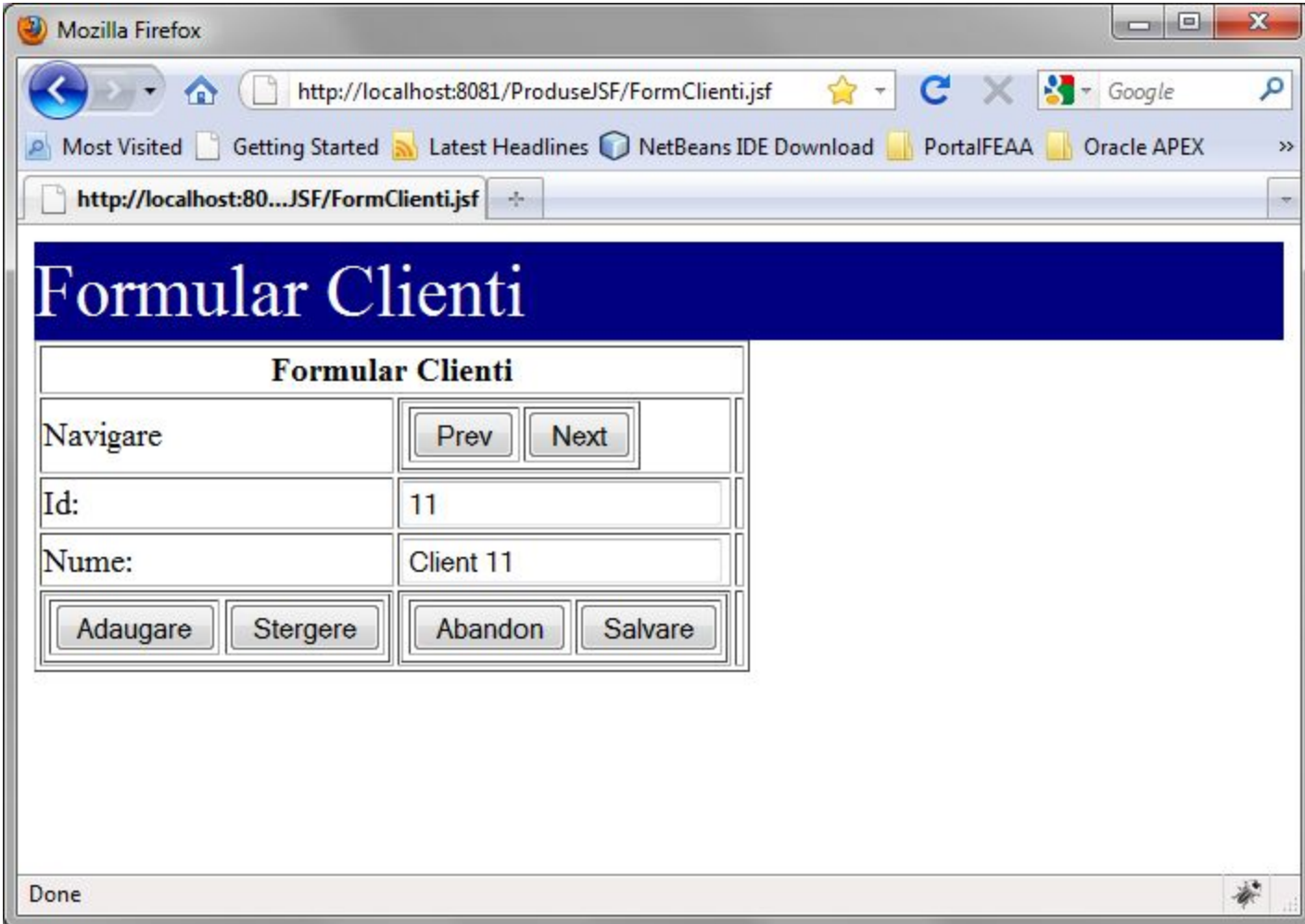
# Adăugare butoane de comandă în formular

---

- *Al doilea panou:*
  - buton pentru abandon:
    - *Id:* `cmdAbandon`
    - *Value:* `Abandon`
    - *Action Listener:* `#{formClienti.abandonClient}`
  - buton pentru salvare:
    - *Id:* `cmdSalvare`
    - *Value:* `Salvare`
    - *Action Listener:* `#{formClienti.salvareClient}`

# Noua “față” a formularului

---



The screenshot shows a Mozilla Firefox browser window with the address bar displaying `http://localhost:8081/ProduseJSF/FormClienti.jsf`. The page title is "Formular Clienti". The form itself is titled "Formular Clienti" and contains the following elements:

Formular Clienti	
Navigare	<input type="button" value="Prev"/> <input type="button" value="Next"/>
Id:	<input type="text" value="11"/>
Nume:	<input type="text" value="Client 11"/>
<input type="button" value="Adaugare"/> <input type="button" value="Stergere"/>	<input type="button" value="Abandon"/> <input type="button" value="Salvare"/>

The status bar at the bottom of the browser window shows "Done".

## 2. Navigare prin selecție în listă-combinată a entității curente

---

- Reutilizare **model de date** suport din clasa back-bean.
  - Adăugare componentă *SelectOneMenu* (echivalentul listei combinate) și legare la element curent din formular.
  - Adăugare specificații pentru popularea listei combinate.
  - Adăugare comportament *ajax*.
-

# Modul de functionare a listei de navigare-selectie

---

- Se bazează pe următoarele repere:
  - popularea listei (de navigare) se va face pe baza colecției de entități din modelul de date al formularului;
  - lista va fi sincronizată cu entitatea curentă a formularului, deci:
    - schimbarea elementului selectat (modificarea valorii listei de selecție) va produce schimbarea entității curente din *modelul* de date;
    - schimbarea entității curente din *modelul* de date (altfel decât prin listă) va produce schimbarea elementului curent al *listei*.



# Modelul de date suport pentru lista de selecție (navigare) din clasa controller-bean

---

- Din punct de vedere logic, o listă JSF (gen *SelectOneMenu* sau *SelectOneListBox*) poate fi populată cu o colecție de entități ale *modelului* de date.
- La nivelul formularului HTML din browser, de fapt, lista va fi formată dintr-o colecție de elemente.

<string de prezentare>-<string valoare>.

- Modificarea elementului curent selectat al listei de navigare este echivalentă cu modificarea valorii listei.
  - Prin urmare, lista va fi legată, din punctul de vedere al valorii, la o proprietate, accesibilă prin operații getter/setter, găzduită de clasa controller.
-

# Lista de navigare a formularului pentru clienti

---

- Va fi alimentata direct din colectia de *clienti* de la nivelul form-beanului.

`List<Client> getClients()`

- Elementele din setul de selectie vor fi formate din perechi de forma nume client-identificator client

`[eticheta] nume - [valoare] id`

- Valoarea listei (de fapt a elementului curent selectat) va fi, prin urmare, identificatorul clientului (`idClient`).
- Valoarea listei va fi legata, prin urmare, la proprietatea `idClient`, desemnata prin getter/setter:

`Integer getIdClient()`

`void setIdClient()`

---

# Modelul de date suport pentru lista de selecție (navigare) clienti

---

```
public class FormClienti implements Converter, Validator {  
  
    private List<Client> clienti = new ArrayList<Client>();  
    private Client client;  
    private EntityManager em;  
  
    public List<Client> getClienti() {  
        return this.clienti;  
    }  
  
    public Integer getIdClient(){  
        return this.client.getId();  
    }  
  
    public void setIdClient(Integer id){  
        Integer idx = this.clienti.indexOf(new Client(id, ""));  
        this.client = this.clienti.get(idx);  
    }  
  
    /* Implementare suport pentru navigare selectie lista combinata */  
}
```

## Adăugare componentă-view listă navigare: *SelectOneMenu* (sau *SelectOneListBox*)

---

- Lista combinată pentru selecție-navigare va fi obținută prin configurarea componentei grafice *SelectOneMenu* din secțiunea *JSF HTML* a paletei.
  - Această componentă va fi adăugată inițial pe prima linie a panoului principal al formularului (**vezi pagina următoare**), după eticheta *Navigare* care va fi eliminată ulterior, astfel încât lista-combinată să o înlocuiască.
  - Lista combinată va fi *legată* la entitatea curentă a formularului.
-

# Formular Clienti

Formular Clienti

Navigare

Prev

Next

message

New Select One Menu s1

Select One Menu s1

Create a new select one menu s1 tag

General

Id: cboClienti

Binding:

Value: #{formClienti.idClient}

Rendered: ☒

Required: ☐

Finish

Cancel

Palette

Output Link

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

Expression Builder

Expression Builder

Build an EL expression that will be evaluated at runtime.

Expression:

#{formClienti.idClient}

Variables:

type filter text

Managed Beans  
formClienti  
client  
clienti  
clientList  
idClient  
selected

Operators:

&&  
||  
>  
<  
>=  
<=  
==  
!=  
!  
+  
-  
\*  
/  
%

OK

Cancel

După **ștergerea** etichetei *Navigare* inițiale, formularul ar trebui să arate astfel:

---

## Formular Clienti

Formular Clienti		
<input type="text"/>	<input type="button" value="Prev"/> <input type="button" value="Next"/>	 message
Id:	<input type="text" value="#{formClienti.client.id}"/>	 message
Nume:	<input type="text" value="#{formClienti.client.nume}"/>	 message
<input type="button" value="Adauga"/> <input type="button" value="Stergere"/>	<input type="button" value="Abandon"/> <input type="button" value="Salvare"/>	 message

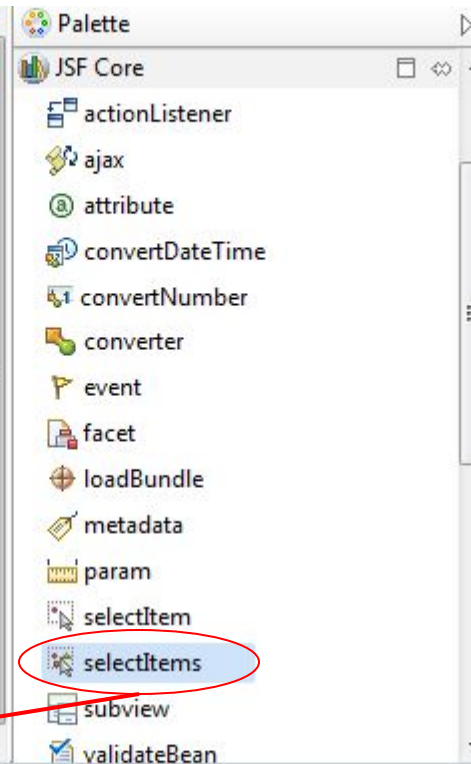
# Adăugare specificații pentru popularea listei combinate

---

- Legarea listei combinate de sursa de date se face prin intermediul componentei non-grafice *SelectedItems* din secțiunea *JSF Core* a paletei. Această componentă poate fi adusă prin *drag&drop* exact între tagurile *<selectOneMenu>...</selectOneMenu>* ce consemnează definiția listei combinate în fereastra cod-sursă aferentă fișierului *xhtml*.
- Înscrierea cursorului de scriere între tagurile *<selectedItems>* nou generate va activa fereastra de proprietăți de unde se va selecta proprietatea *clienti* a beanului-controller *formClienti*, **după cum se exemplifică în pagina următoare.**

# Formular Clienti

Formular Clienti			
<input type="text"/>	<input type="button" value="Prev"/> <input type="button" value="Next"/>		
Id:	<input type="text" value="#{formClienti.client.id}"/>		
Nume:	<input type="text" value="#{formClienti.client.nume}"/>		
<input type="button" value="Adauga"/> <input type="button" value="Stergere"/>	<input type="button" value="Abandon"/> <input type="button" value="Salvare"/>		



Design Preview

Problems Servers Properties

General

Select Items

Value:

Expression Builder

Build an EL expression that will be evaluated at runtime.

Expression:

Variables:

- Managed Beans
  - formClienti
    - client
    - clienti
    - clientList
    - idClient
    - selected

Operators:

- &&
- >
- <
- >=
- <=
- ==
- !=
- !
- +
- 
- \*
- /
- %

OK Cancel




# Adăugare specificații pentru popularea listei combinate

---

- Pentru desemnarea modului cum entitățile din sursa de date, proprietatea `clienti` accesată prin `getClienti()`, sunt convertite în perechi eticheta-valoare de forma `nume-id`, componenta (non-grafică) *selectedItems* va fi configurată prin proprietățile:
  - **var** cu valoarea **item**:
    - ca urmare fiecare client va fi accesat la iterarea sursei de date (proprietatea `value`) prin variabila cu numele `item`;
  - **itemLabel** cu valoarea `#{item.nume}`
  - **itemValue** cu valoarea `#{item.id}`

# FormClienti.2.0

## Formulari Clienti

<input type="text"/>	<div>PrevNext</div>	 message
ID Clienti	<input type="text" value="#{formClienti.client.id}"/>	 message

```
<h:form id="formClientiForm">
  <h:panelGrid columns="3" border="1">
    <p:facet name="header">
      <h:outputText value="Formulari Clienti"></h:outputText>
    </p:facet>

    <h:selectOneMenu id="cboClienti" value="#{formClienti.idClient}" style="width: 209px; ">
      <p:selectItems value="#{formClienti.clienti}"
        var="item"
        itemValue="#{item.id}"
        itemLabel="#{item.nume}">
      </p:selectItems>
      <p:ajax render="id nume"></p:ajax>
    </h:selectOneMenu>
  </h:panelGrid>
</h:form>
```

Design Preview

Markers Properties Servers Data Source Explorer Snippets Console REST Annotations WLST Help

### selectItems

All	Property	Value
	Attributes	
	binding	
	id	
	itemDescription	
	itemDisabled	
	itemLabel	#{item.nume}
	itemLabelEscaped	
	itemValue	#{item.id}
	value	#{formClienti.clienti}
	var	item

# Adăugare comportament *ajax*

---

- Pentru ca lista combinată să aibă un comportament corespunzător, e nevoie de adăugarea unor specificații *ajax* astfel încât:
- ◆ comportamentul listei (mai exact schimbarea elementului curent al listei) să nu retrimită pe server decât noua stare a listei, fără celelalte componente ale formularului;
  - ◆ după procesarea pe server a listei, ce se va traduce prin apelarea *setterului* *setIdClient*, pentru a desemna drept entitate nou selectată în formular pe cea corespunzătoare valorii listei, vor trebui *reuate* de pe server doar valorile corespunzătoare rubricilor simple (*refresh* pe rubricile care vor prezenta proprietățile entității nou selectate, adică rubricile cu id-urile *id* și *nume*).
-

# Adăugare comportament *ajax*

---

- Acest comportament se face specifica astfel:
  - din secțiunea *JSF Core* a paletei se aduce (drag) componenta non-vizuală *Ajax* în contextul formularului, și se plasează, la fel ca și *selectedItem*, tot între tagurile `<selectOneMenu>...</selectOneMenu>` ce consemnează definiția listei combinate în fereastra cod-sursă aferentă fișierului *xhtml* (vezi pagina următoare);
  - după înscrierea cursorului de scriere între tagurile `<ajax>` nou generate se vor menționa **id-urile rubricilor simple** ale formularului (despărțite obligatoriu prin **spațiu**, și nu prin virgulă) în proprietatea *render* din fereastra de proprietăți asociată tagului *ajax*.

FormClienti.java FormClienti2.xhtml Client.java

**formClienti**

Prev Next message

Nume: #{formClienti.client.nume} message

Id: #{formClienti.client.id} message

Adauga Sterge Abandon Salveaza message

```
13 </p:facet>
14
15 <h:selectOneMenu id="cboClienti" value="#{formClienti.
16 <p:selectItems id="s1" value="#{formClienti.client
17 <p:ajax render="client_id client_nume"></p:ajax>
18 </h:selectOneMenu>
19 <h:panelGrid border="1" columns="2" id="p2">
20 <h:commandButton id="cmdPrevious" value="Prev" act
21 <h:commandButton id="cmdNext" value="Next" actionL
22 </h:panelGrid>
23 <h:message id="m3"></h:message>
24 <h:outputText id="o2" value="Nume:"></h:outputText>
25 <h:inputText id="client_nume" value="#{formClienti.cl
```

Design Preview

Markers Properties Servers Data Source Explorer Snippets Console

**Ajax Type false**

General

Render: client\_id client\_nume

Event:

Execute:

Immediate: ☐

Listener:

JavaScript

On event:

On error:

Finish Cancel

**New Ajax Type false**

Ajax Type false

Create a new ajax type false tag

General

Render: client\_id client\_nume

Event:

Execute:

Immediate: ☐

Listener:

JavaScript

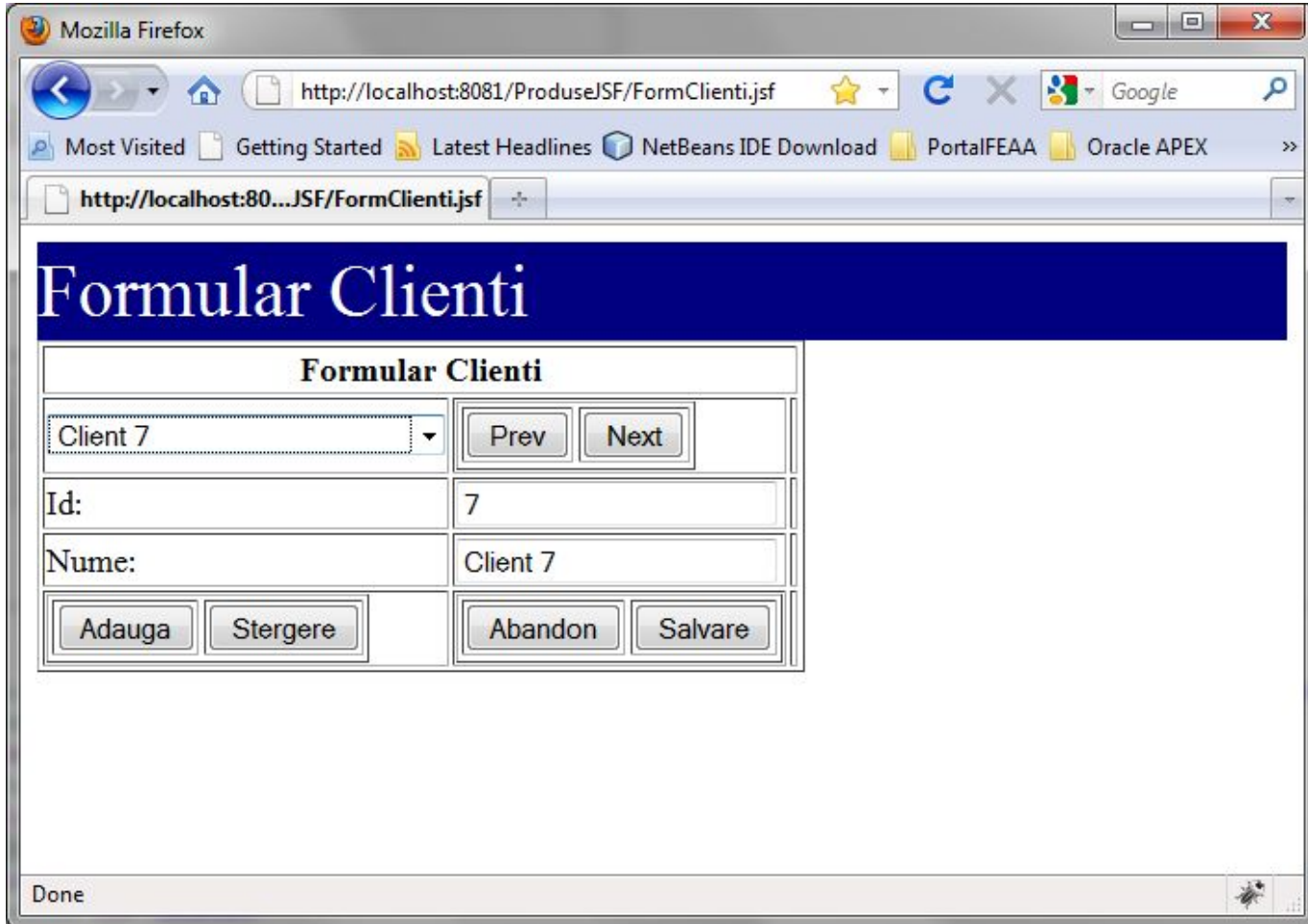
On event:

On error:

Finish Cancel

### 3. În final, formularul ar trebui să arate astfel

---



The screenshot shows a Mozilla Firefox browser window with the address bar displaying `http://localhost:8081/ProduseJSF/FormClienti.jsf`. The browser's toolbar includes navigation buttons, a search bar with the Google logo, and a list of bookmarks: 'Most Visited', 'Getting Started', 'Latest Headlines', 'NetBeans IDE Download', 'PortalFEAA', and 'Oracle APEX'. The main content area features a dark blue header with the text 'Formular Clienti' in white. Below the header, the form is titled 'Formular Clienti' in bold. It contains a dropdown menu with 'Client 7' selected, and two buttons labeled 'Prev' and 'Next'. Below these are two rows of input fields: 'Id:' with the value '7', and 'Nume:' with the value 'Client 7'. At the bottom of the form are four buttons: 'Adauga', 'Stergere', 'Abandon', and 'Salvare'. The status bar at the bottom of the browser window shows 'Done'.

Formular Clienti	
Client 7	Prev Next
Id:	7
Nume:	Client 7
Adauga Stergere	Abandon Salvare