Primjena ADF Swing tehnologije

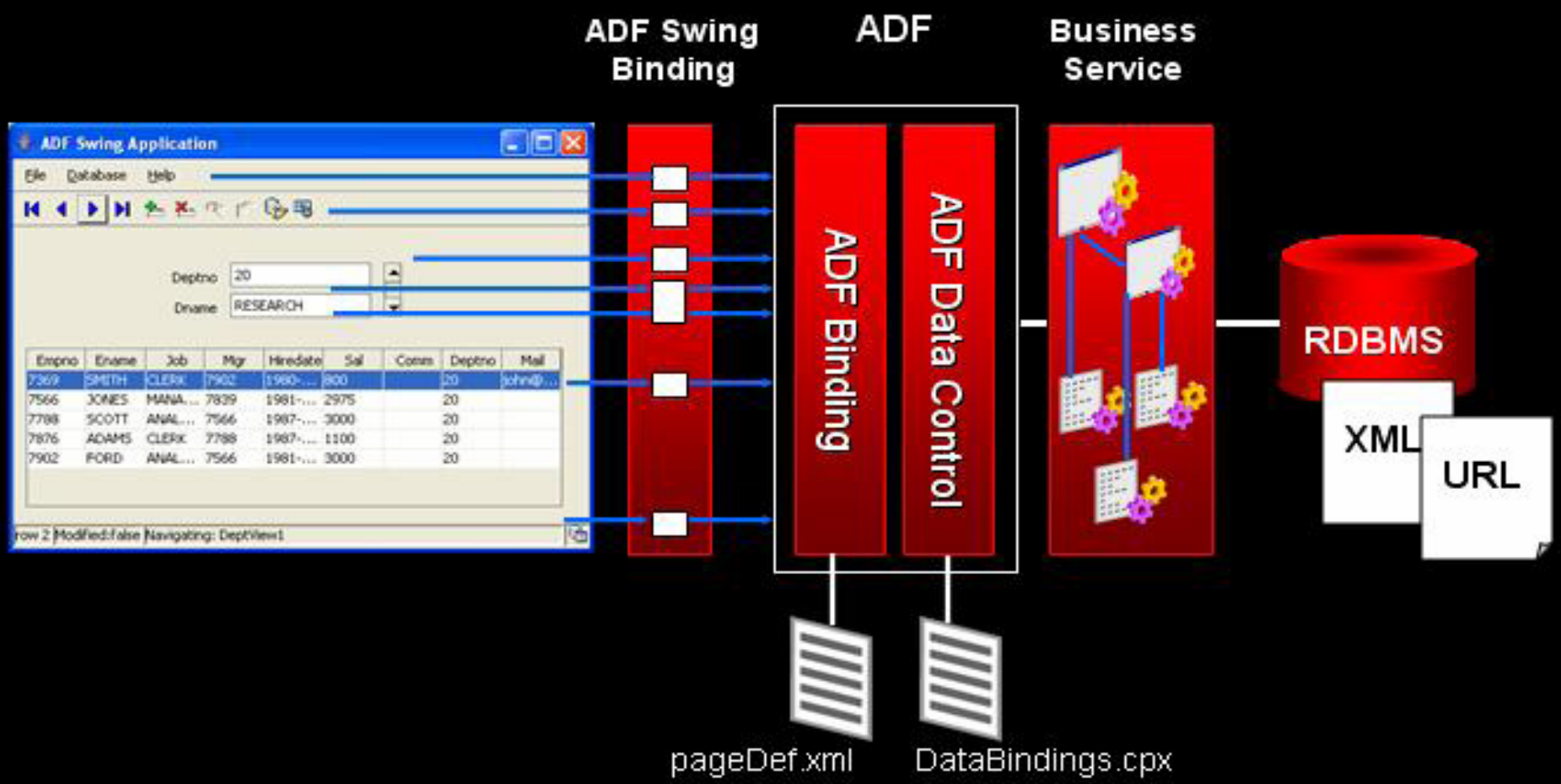
Andreja Migles
NEOS d.o.o.

Sadržaj

- Komponente ADF Swing-a
- ADF Swing UI komponente
- Pristup metapodacima iz Java koda
- Primjer iz prakse

Komponente razvoja ADF Swing aplikacije

- JDeveloper
- Oracle ADF
  - Data Controls
  - Iterator Bindings
  - Binding Containers
  - Binding Context
- SWING


ADF Swing Binding
ADF
Business Service
ADF Swing Application
File
Database
Help
Deptno
20
Dname
RESEARCH
Empno
Ename
Job
Mgr
Hiredate
Sal
Comm
Deptno
Mail
7369
SMITH
CLERK
7902
1980-...
800
20
7566
JONES
MANA...
7839
1981-...
2975
20
7788
SCOTT
ANAL...
7566
1987-...
3000
20
7876
ADAMS
CLERK
7788
1987-...
1100
20
7902
FORD
ANAL...
7566
1981-...
3000
20
row 2
Modified:false
Navigating: DeptView1
ADF Binding
ADF Data Control
RDBMS
XML
URL
pageDef.xml
DataBindings.cpx

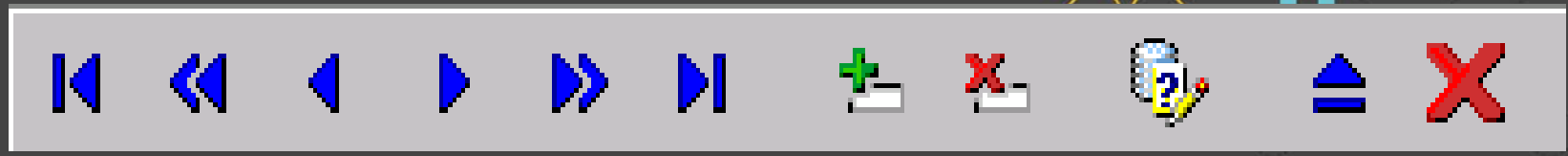
pageDef.xml

```
<executables>
  <iterator id="UrediViewIterator" RangeSize="10" Binds="UrediView"
            DataControl="AppModuleDataControl"/>
  <iterator id="UrediPosteViewIterator" RangeSize="10" Binds="UrediPosteView"
            DataControl="AppModuleDataControl"/>
</executables>
<bindings>
  <attributeValues id="UrediViewSifraUreda" IterBinding="UrediViewIterator"
                   xmlns="http://xmlns.oracle.com/adfm/jcuimodel">
    <AttrNames xmlns="http://xmlns.oracle.com/adfm/uimodel">
      <Item Value="SifraUreda"/>
    </AttrNames>
  </attributeValues>
```

ADF Swing UI komponente

- BIGraph
- JULabel
- JUNavigationBar
- JUStatusBar
- JURadioButtonGroupPanel
- JUImageControl
- JUArrayComboBox

JUNavigationBar



- način povezivanja sa iteratorom:
  jUNavigationBar1.setModel(JUNavigationBar.createViewBinding(panelBinding,
  jUNavigationBar1, "UrediView", null, "UrediViewIterator"));
- ako želimo određene akcije nadjačati potrebno je kreirati klasu koja nasljeđuje JUNavigationBar i nadjačati metodu doAction

```
if (action ==BUTTON_COMMIT){
  System.out.println("Akcija prije commit");
}
super.doAction(action);
```

JUNavigationBar

- Moguće je promijeniti ikone navigacijskog bara
- Ikone se postavljaju u metodi

  _updateButtonStates() i _updateButtonStatesLater()
- ***this.getButton(this.BUTTON_INSERT).setIcon(new ImageIcon(Util.getImageAsResource(APPSettings.BUTTON_ADD)));***

JUStatusBar

- Informacije o ukupnom broju slogova, trenutačnom slogu….

```
public class CustomStatusBar extends JUStatusBar{
    public CustomStatusBar() {

    this.setHasPercentDone(true);
    this.setHasRowCount(true);

    this.setRowCountFormatString("  " +   "######" + " rows  ");
    this.setCurrentRowFormatString(" Row " +   "######" + " ");
```

Pristup metapodacima iz Java koda

- Svaki panel sadrži JUPanelBinding objekt pomoću kojeg pristupamo ADF binding container-u

```
JUPanelBinding panelBinding = new JUPanelBinding("UredPanelPageDef");
DCBindingContainer bc = getPanelBinding().getBindingContainer();
DCBindingContainer bc = getPanelBinding().getBindingContainer();
DCIteratorBinding iter = bc.findIteratorBinding("MyIter");

jTxtFldNazivPoste.setColumns((panelBinding.findCtrlValueBinding("Ured
PosteViewNazivPoste")).getDisplayWidth());

jTxtFldSifraUreda.setDocument((Document)panelBinding.bindUIControl("Ur
    ediViewSifraUreda",jTxtFldSifraUreda));
```

Pristup metapodacima iz Java koda

- DCBindingContainer bc = getPanelBinding().getBindingContainer();
  JUCtrlActionBinding ab = (JUCtrlActionBinding) bc.findCtrlBinding("MyActionBinding");
  ab.invoke();

- DemoModule dm = (DemoModule)getPanelBinding()
  .getBindingContext()
  .findDataControl("YourDataControl")
  .getDataProvider();
  dm.yourCustomMethod(your,args);

Primjer iz prakse

- Svaki panel ima vlastiti BindingContext tj. vlastiti session što se pridružuje u kontruktoru svakog panela
- Otvaranjem panela otvara se nova konekcija nad bazom
- Zatvarenjem određenog panela konekcija ostaje otvorena, ali se pridruzuje nekom drugom panelu

Dohvaćanje konekcije iz pool-a

```
public OsobaPanel() {
    JUMetaObjectManager.setErrorHandler(new JUErrorHandlerDlg());
    JUMetaObjectManager mgr = JUMetaObjectManager.getJUMom();
    mgr.setJClientDefFactory(null);
    BindingContext ctxl = new BindingContext();
    ctxl.put(DataControlFactory.APP_PARAM_ENV_INFO, new JUEnvInfoProvider
    ctxl.setLocaleContext(new DefLocaleContext(null));
    HashMap map = new HashMap(4);
    map.put(DataControlFactory.APP_PARAMS_BINDING_CONTEXT, ctxl);
    mgr.loadCpx("hr.neos.pis.view.DataBindings.cpx" , map);
    this.setBindingContext(ctxl);
    this.revalidate();
}
```

Izvršavanje dodatnih operacija prije unosa, brisanja...

- Svaki entitet nasljeđuje klasu **CustomEntityImpl extends EntityImpl**
- U klasi CustomEnitityImpl nadjačana je metoda doDML u kojoj se prije unosa izvršavaju unosi u dodatne kontrolne tablice

```
protected void doDML(int i, TransactionEvent transactionEvent)  {
 try{
    if ( (i ==DML_INSERT ||  i == DML_UPDATE) && this.getTableName() != null){
```

Dinamičko dohvaćanje tipa, naziva, vrijednosti atributa

```
AppModuleImpl appModule = (AppModuleImpl)this.getDBTransaction().getRootAppli
// punim tablicu ULAZ
Row r =  appModule .getUlaz().createRow();
```

```
String[] s =this.getAttributeNames();
```

```
if(this.getEntityDef().getAttributeDefImpl(s[ib]).getSQLType() ==12){
    tip = "V";
}
rArgl.setAttribute("TipArgumenta",tip);
rArgl.setAttribute("VrijednostArgumenta",this.getAttribute(
this.getEntityDef().getAttributeDefImpl(s[ib]).getIndex()));
rArgl.setAttribute("NazivArgumenta",
this.getEntityDef().getAttributeDefImpl(s[ib]).getColumnName());
```

Sortiranje podataka u tablici

- Potrebno je u projekt uključiti izvorni kod klase

  oracle.jbo.uicli.jui. JUTableBinding

  i promijeniti metodu ***getControlModel*** kako bi se sortiranje izvršilo na željeni način

Jednaka boja za sva obavezna polja 1

- Sva polja su instance klase

CustomInputTextField extends JTextField

- Klasa sadrži 2 atributa kojima se pridružuje vrijednost prilikom kreiranja svakog polja:

// naziv xml atributa na koji je povezano polje

– *private String bindingColumn*;

//xml na koji je povezano polje

– *private JUPanelBinding columnBindingPanel*;

Jednaka boja za sva obavezna polja 2

- Na temelju objekta JUPanelBinding dolazi se do informacije da li je polje obavezno te mu se ovisno o tome postavlja boja:

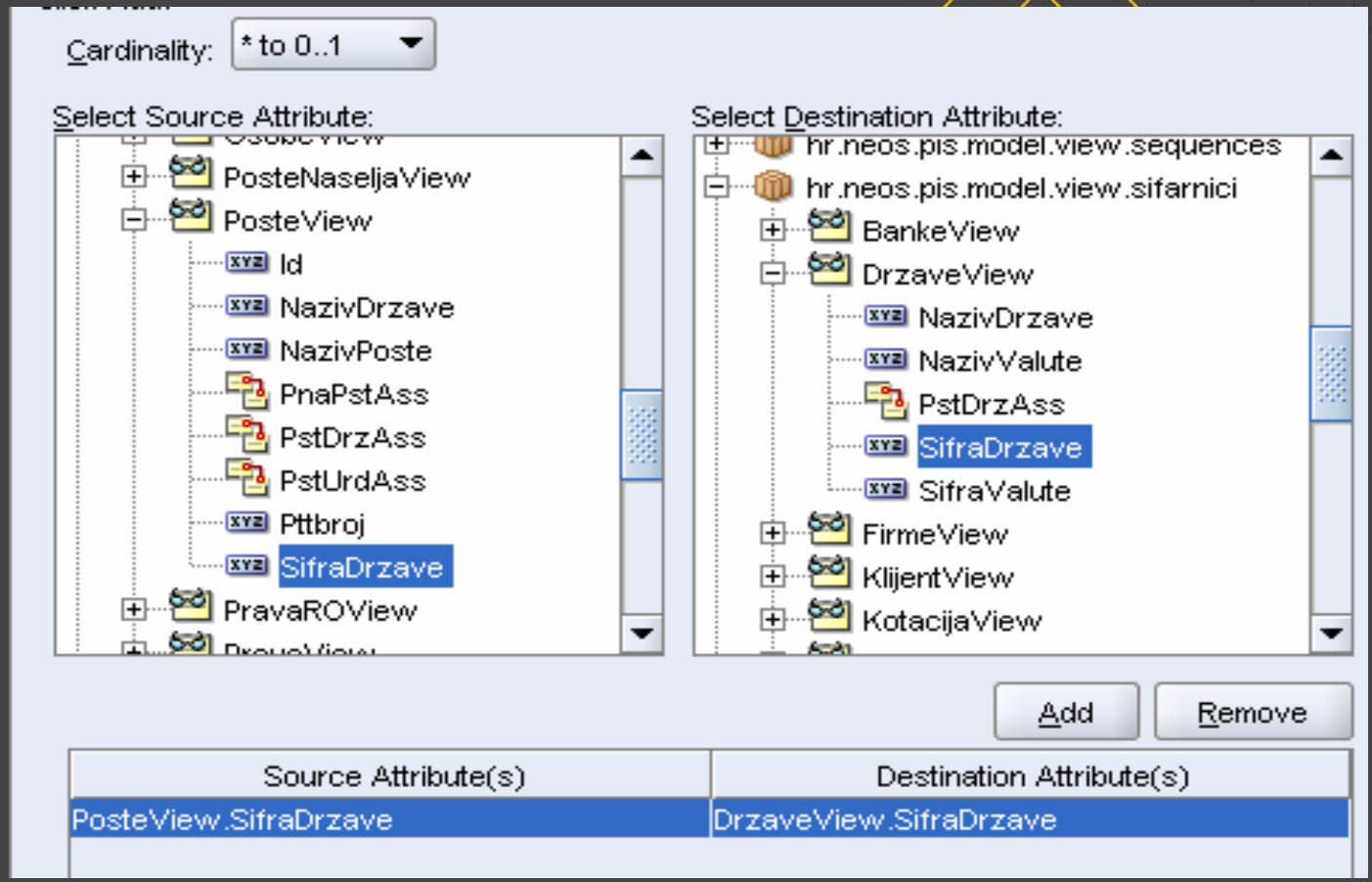
columnBindingPanel.findCtrlValueBinding (bindingColumn)).isMandatory();

Dinamička promjena atributa iz ReadOnly u Updatable

```
AttributeDefImpl def = (AttributeDefImpl)this.getViewObject().lookupAttributeDef("NazivFirme");
      if (def != null) {
          def.setUpdateableFlag(AttributeDef.UPDATEABLE);
      }
```

```
if (def != null) {
     def.setUpdateableFlag(AttributeDef.READONLY);
}
```

Lookup polja 1



Lookup polja 2

- Najjednostavniji način za prikaz naziva države je uključiti na ekran atribut koji predstavlja naziv iz view-a Drzave jer će se naziv automatski prikazati i prilikom unosa šifre države
- Nedostatak je da nad takvim poljima ne možemo izvršavati pretraživanje
- Rješenje je da se taj atribut uključi u view Poste – na ovaj način neće se puniti naziv države automatski prilikom unosa šifre države već je to potrebno implementirati ručno

Lookup polja 3

```
public void setSifraDrzave(Number value) {
    setAttributeInternal(SIFRADRZAVE, value);
    AttributeDefImpl def = (AttributeDefImpl)this.getViewObject().lookupAttributeDef("NazivDrzave");
    if (def != null) {
     def.setUpdateableFlag(AttributeDef.UPDATEABLE);
    }

    if(this.getDrzaveView()!= null){
       this.setNazivDrzave(this.getDrzaveView().getAttribute("NazivDrzave").toString());
    }else{
        this.setNazivDrzave(null);
    }
```

Liste vrijednosti

- Za svaku listu postoji zaseban panel
- Prilikom poziva, listi se proslijeđuje trenutni iterator i naziv atributa koji je potrebno popuniti nakon odabira sloga na listi
- Odabir vrijednosti na listi:

```
this.iter.getCurrentRow().setAttribute(ps
SifraFirme,jTable1.getModel().getValueAt
(selectedRow,0).toString());
```

Autentikacija korisnika

- Koristi se JAAS - skup API-a i interfacea za izvršenje autentikacije i autorizacije korisnika.
- Za autentikaciju se koriste tzv. LoginModuli koji izvršavaju autentikaciju na temelju informacija callback handlera koji šalju username i password loginmodulu
- Autentikacija na temelju korisnika pohranjenih u baznoj tablici
- Koristi se postojeće riješenje – LoginModuli koji izvršava autentikaciju na temelju određenih tablica

Autentikacija korisnika

- **System-jazn-dana.xml** – definirati koji login modul se koristi te svojstva za autentikaciju korisnika (baza, tablica, driveri..)
- <application>
  <name>CustomDbLoginModule</name>
  <login-modules>
  <login-module>
  <class>hr.neos.security.custom.dbtable.loginmodules.DBTableLoginModule</class>
  <control-flag>required</control-flag>
  <options>
  <option>
  <name>jdbcDriver</name>
  <value>oracle.jdbc.driver.OracleDriver</value>
  </option>

Autentikacija korisnika

- Na razini aplikcijskog modula definirati login modul koji se primjenjuje

jbo.security.enforce	Auth
jbo.security.loginmodule	CustomDbLoginModule

Autentikacija korisnika

- Potrebno je također kreirati login dialog (može se putem wizarda) čija instanca se proslijeđuje kod dohvačanja instance aplikacijskog modula

```
JUMetaObjectManager.setErrorHandler(new JUErrorHandlerDlg());
JUMetaObjectManager mgr = JUMetaObjectManager.getJUMom();
mgr.setJClientDefFactory(null);
BindingContext ctx = new BindingContext();

c = new JCLoginDialog();
c.setCtx(ctx);

ctx.put(DataControlFactory.APP_PARAM_ENV_INFO, c);
```