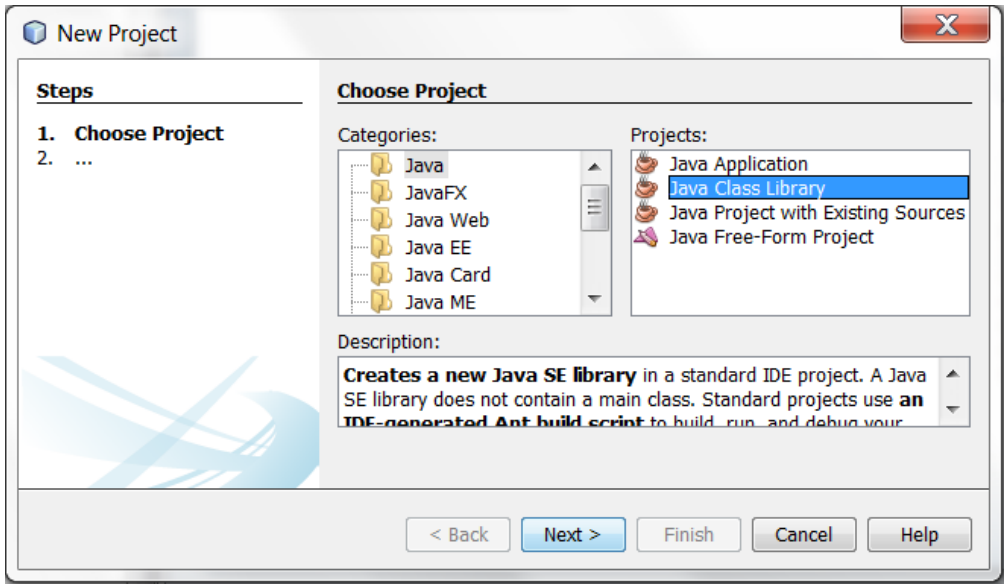
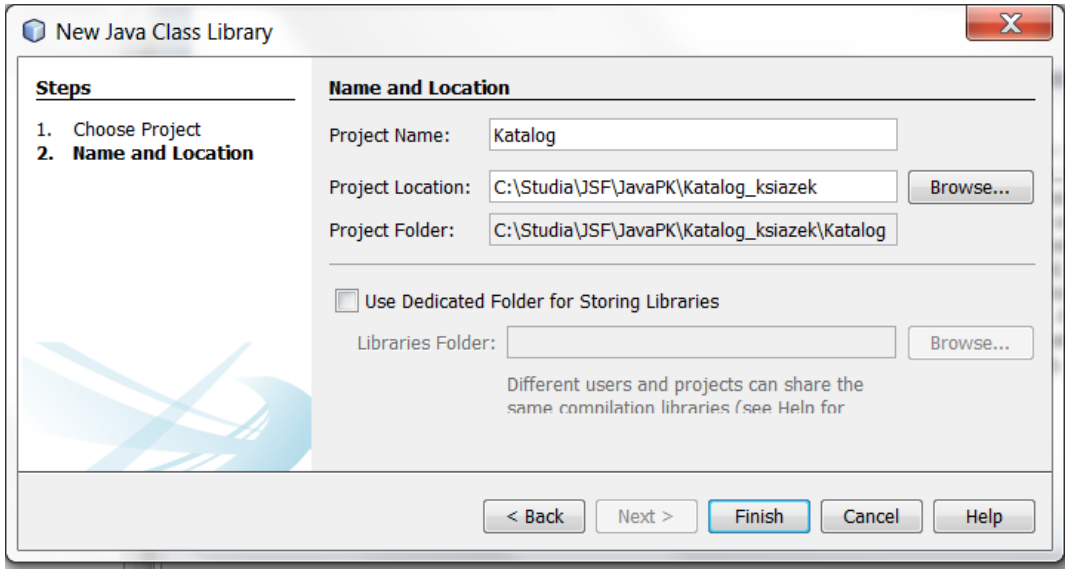
Przekształcanie aplikacji internetowej typu JSF wygenerowanej na podstawie bazy danych do aplikacji typu EE zawierającej dodatkowo klienta typu Enterprise.

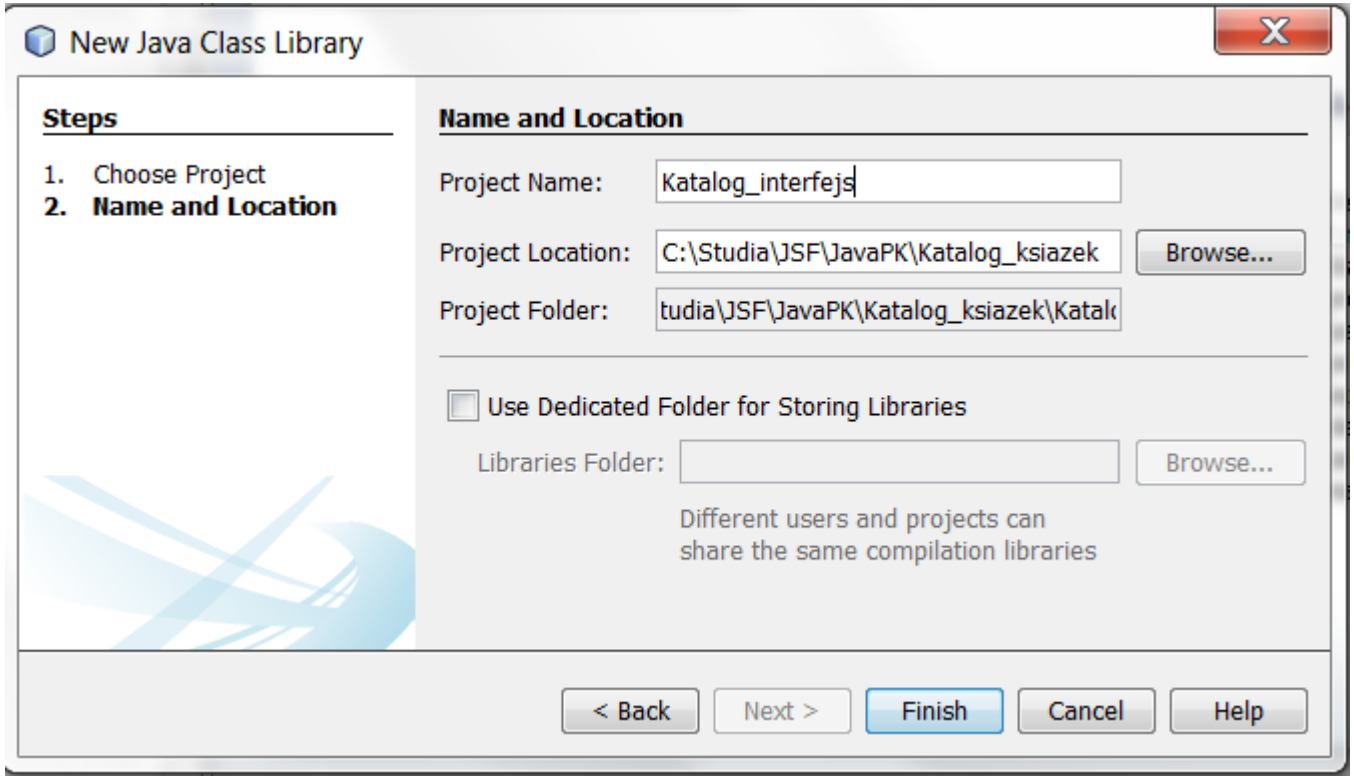
Zofia Kruczkiewicz

1. Zakładanie projektu Katalog typu Java Class Library do przechowywania obiektowego modelu danych – projekt należy do **warstwy biznesowej**

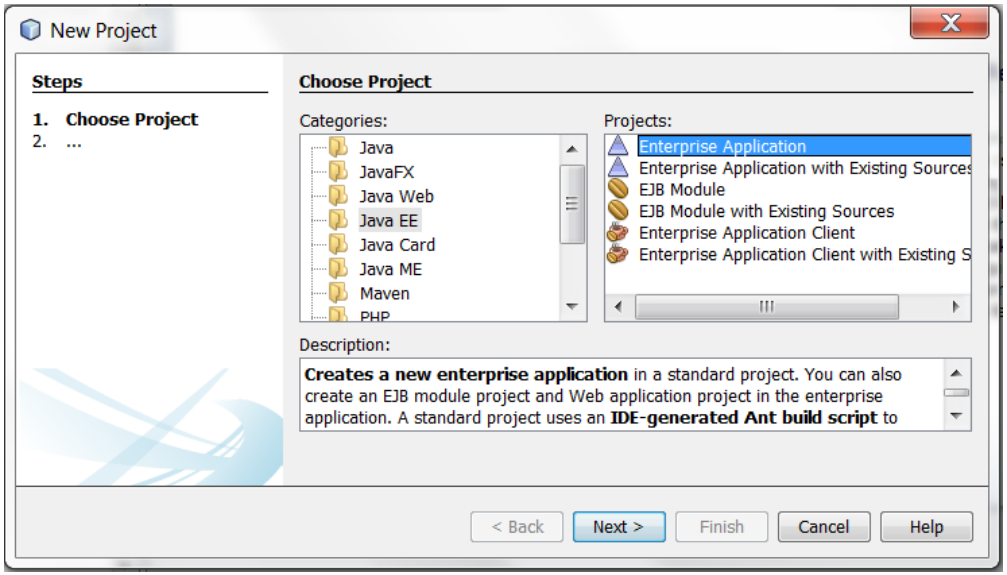


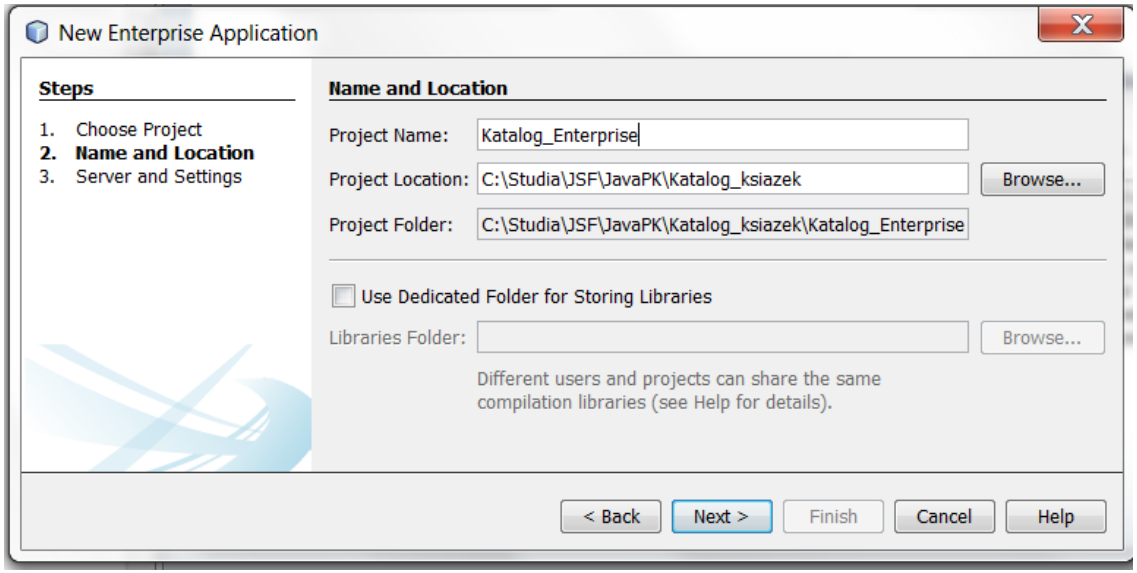


2. Zakładanie projektu Katalog_interfejs typu Java Class Library do przechowywania interfejsów obiektów typu Session Bean for Entity Class (technologia JTA) – projekt należy do **warstwy biznesowej**

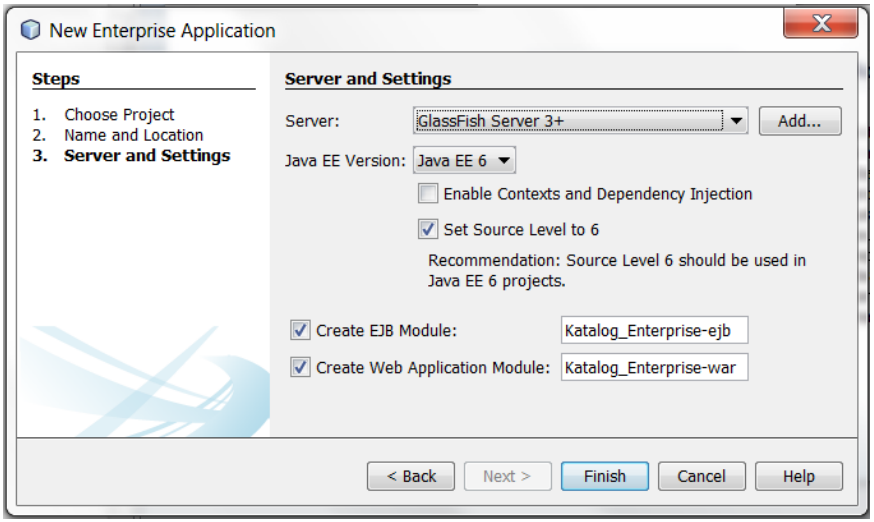


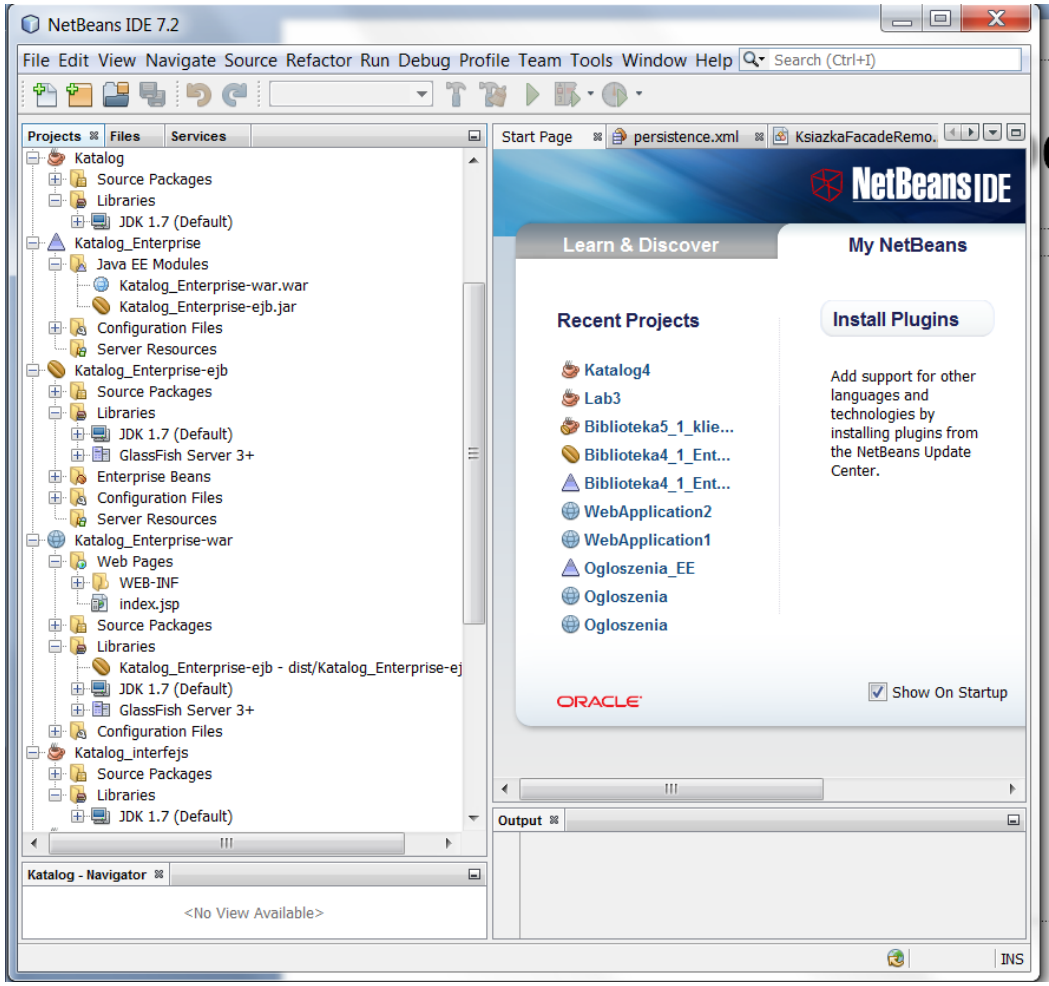
3.1. Zakładanie głównego projektu typu EE – wybór typu aplikacji nadanie nazwy oraz umieszczenie jej w wybranym katalogu. Projekt należy do **warstwy biznesowej.**



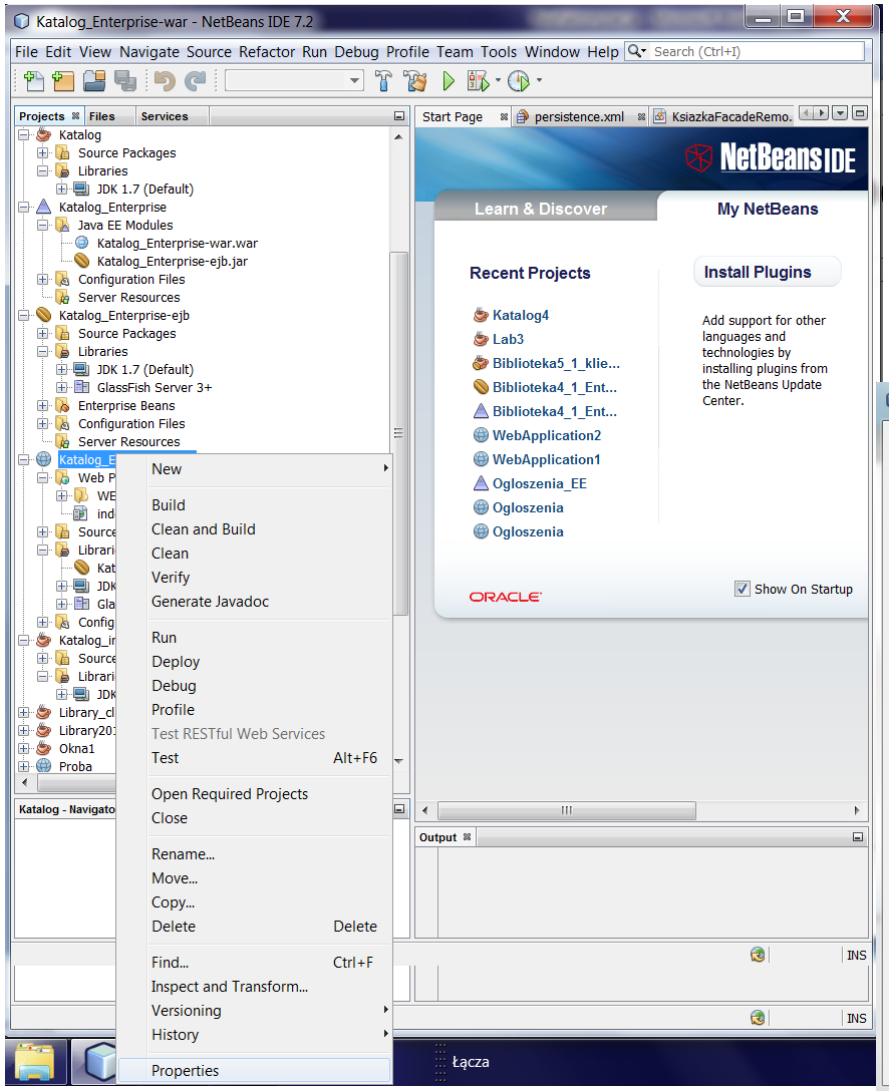


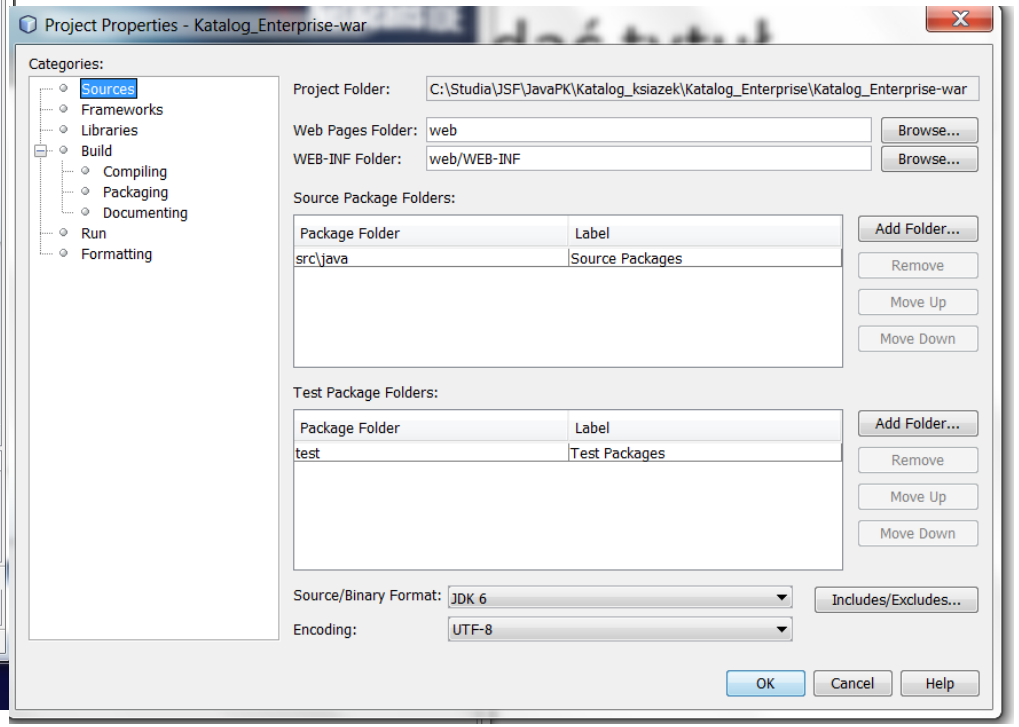
3.2. Wybór serwera aplikacji GlassFish Server 3, platformy Java EE 6 oraz modułu EJB Katalog_Enterprise-ejb oraz modułu internetowego Katalog_Enterprise-war (lewa część strony). Po prawej projekty po zakończeniu czynności ze stron 2-5



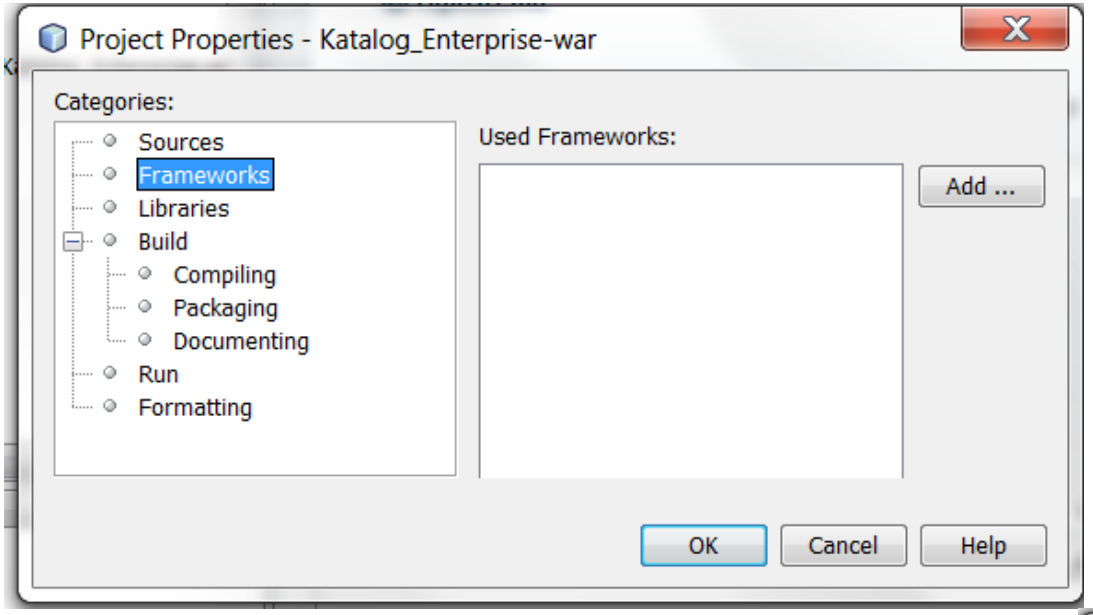


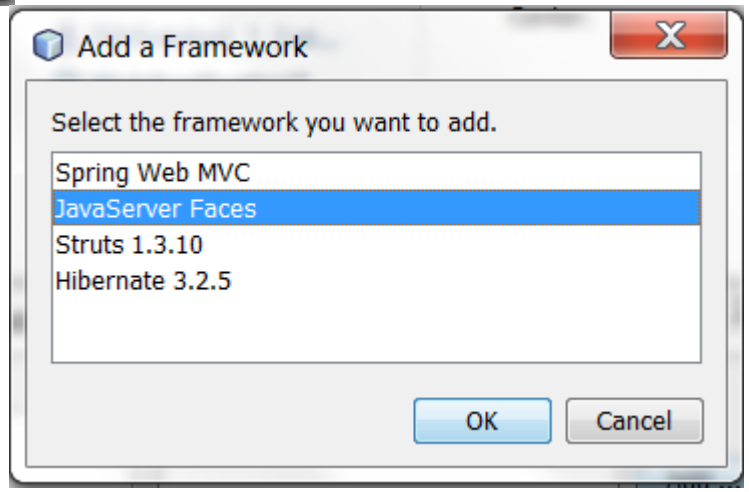
4.1. Zmiana frameworka w module internetowym Katalog_Enterprise-war: po kliknięciu prawym klawiszem myszy nazwę projektu wybór pozycji Properties. Projekt należy **warstwy prezentacji**.



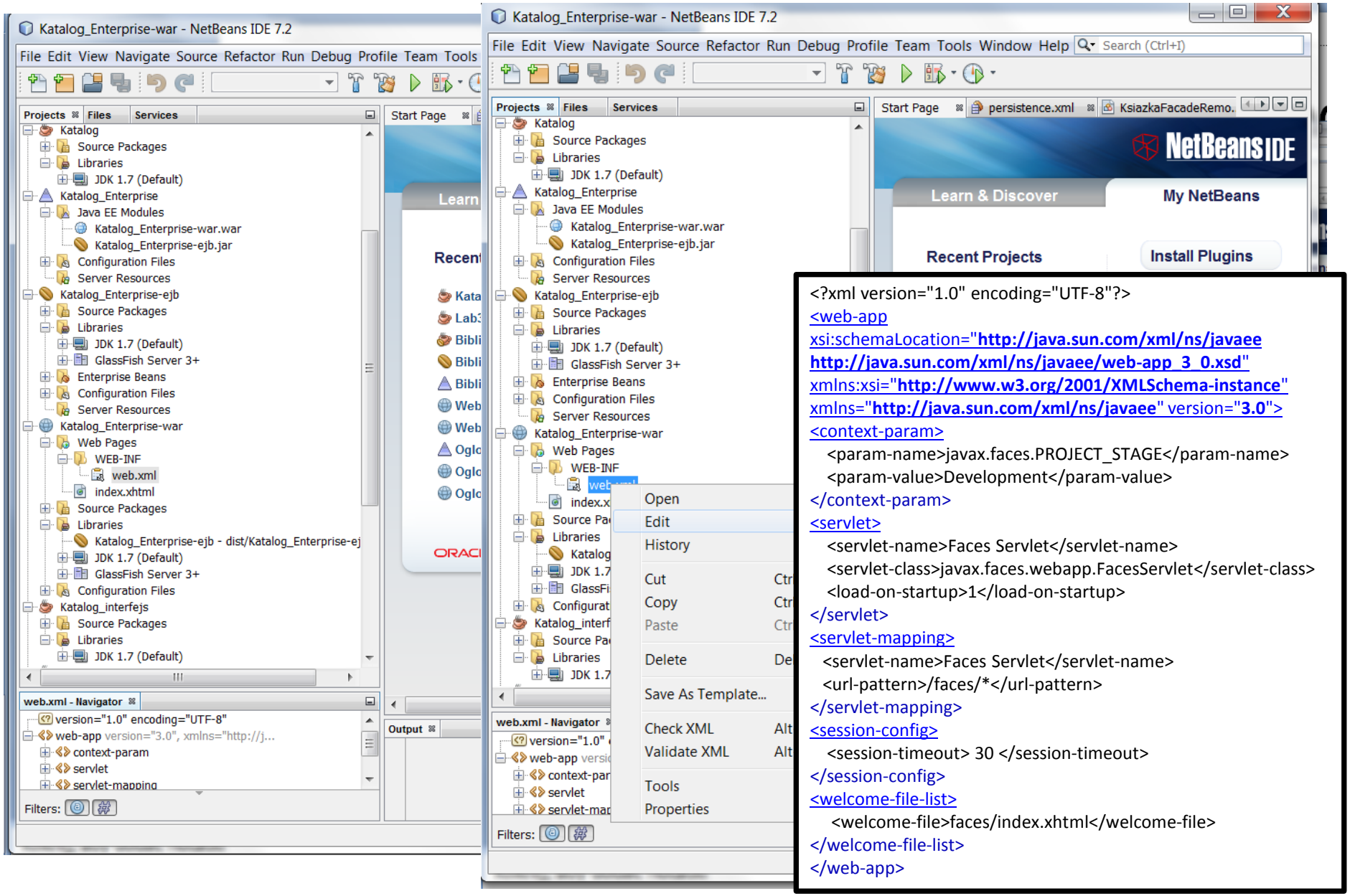


4.2. Zmiana frameworka w module internetowym Katalog_Enterprise-war: wybór pozycji Frameworks (lewa strona) w formularzu Properties oraz kliknięcie na klawisz Add… i wybór w formularzu Add a Framework technologii JavaServer Faces (prawa część strony).



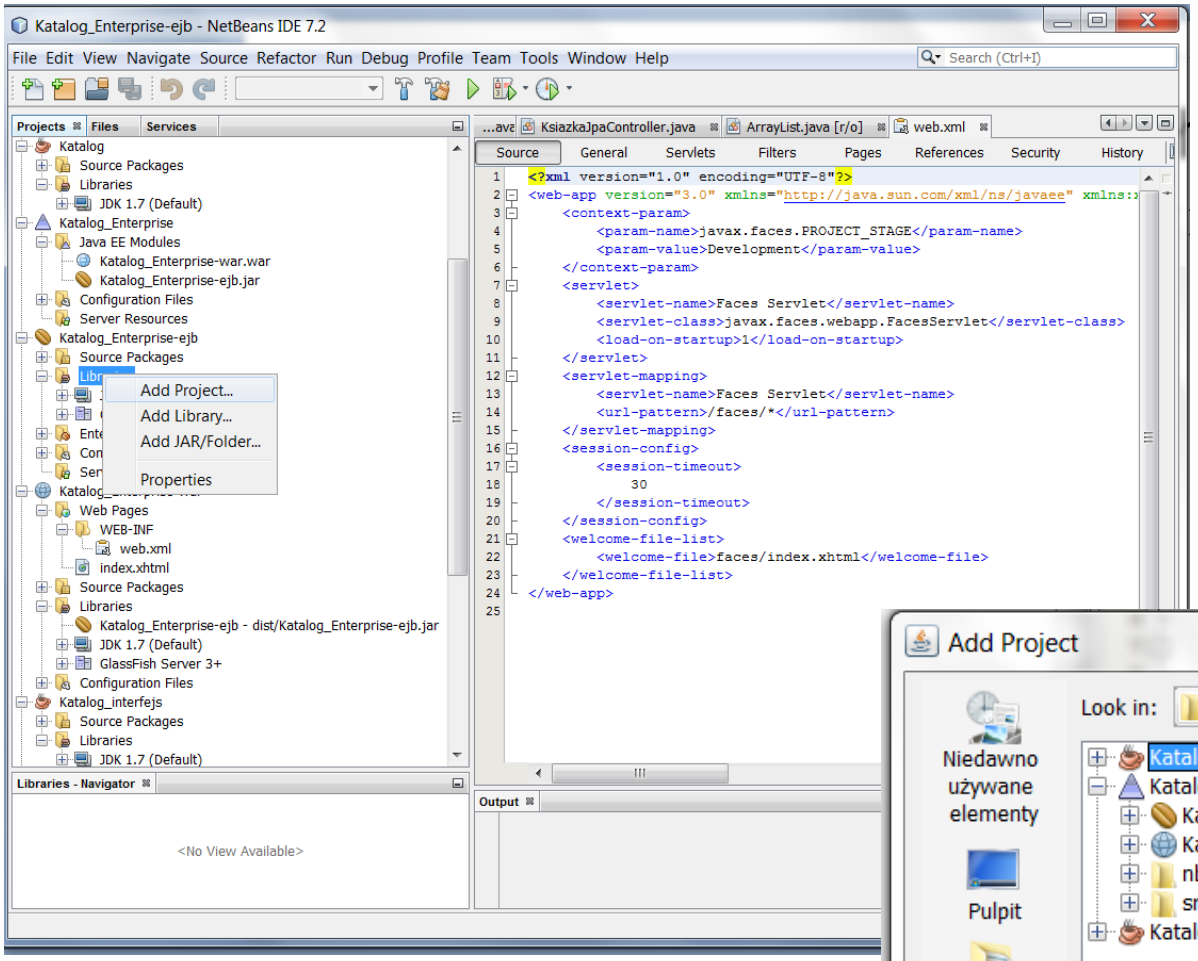


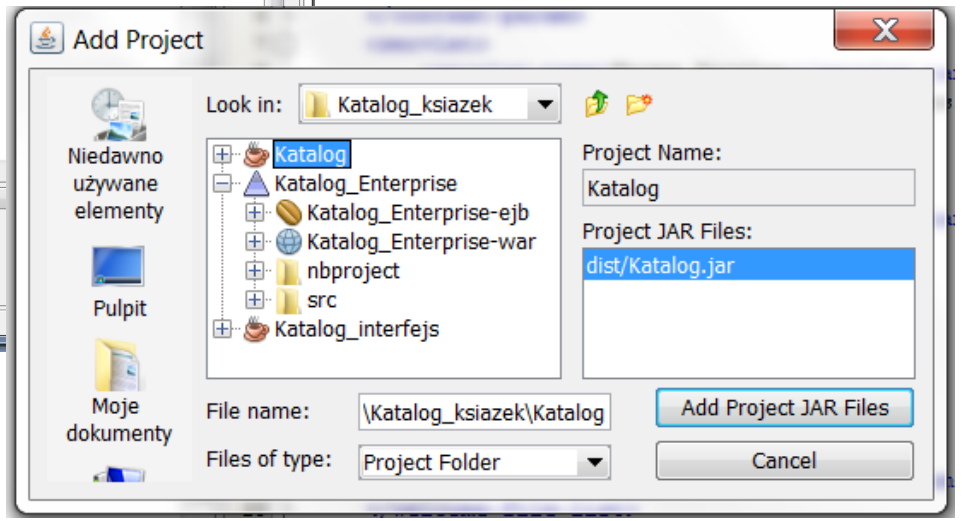
4.3. Sprawdzenie zawartości pliku deskryptora web.xml w module internetowym Katalog_Enterprise-war



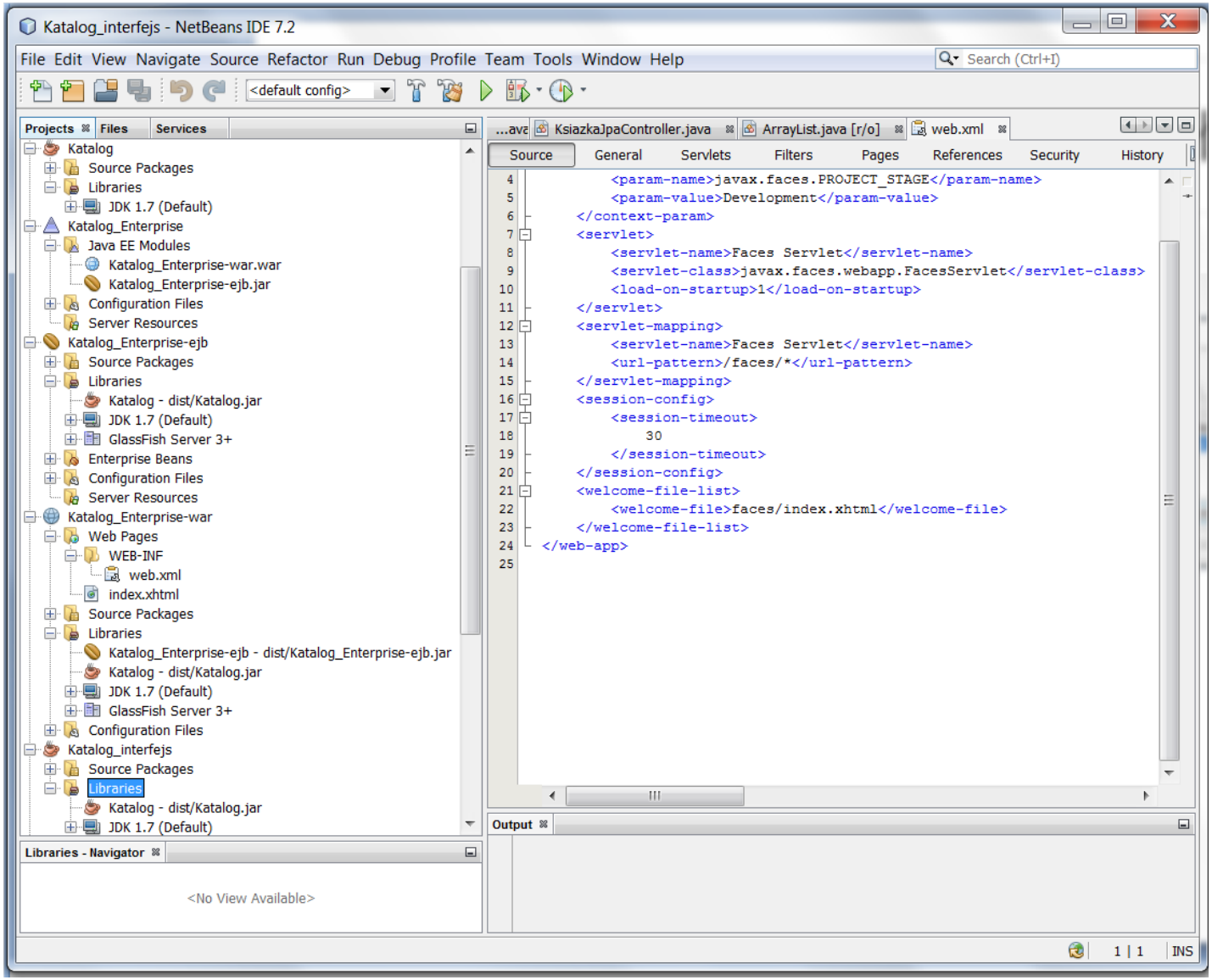

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee" version="3.0">
<context-param>
   <param-name>javax.faces.PROJECT_STAGE</param-name>
   <param-value>Development</param-value>
</context-param>
<servlet>
   <servlet-name>Faces Servlet</servlet-name>
   <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
   <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>/faces/*</url-pattern>
</servlet-mapping>
<session-config>
   <session-timeout> 30 </session-timeout>
</session-config>
<welcome-file-list>
    <welcome-file>faces/index.xhtml</welcome-file>
</welcome-file-list>
</web-app>
```

5.1. Dodanie do katalogów Libraries projektów Katalog_Enterprise-ejb, Katalog_interfejs, Katalog_Enterprise-war **projektu Katalog**: po kliknięciu prawym klawiszem myszy na katalog Libraries wybór pozycji Add Project…, następnie wybór projektu Katalog i naciśnięcie klawisza Add Project JAR Files.

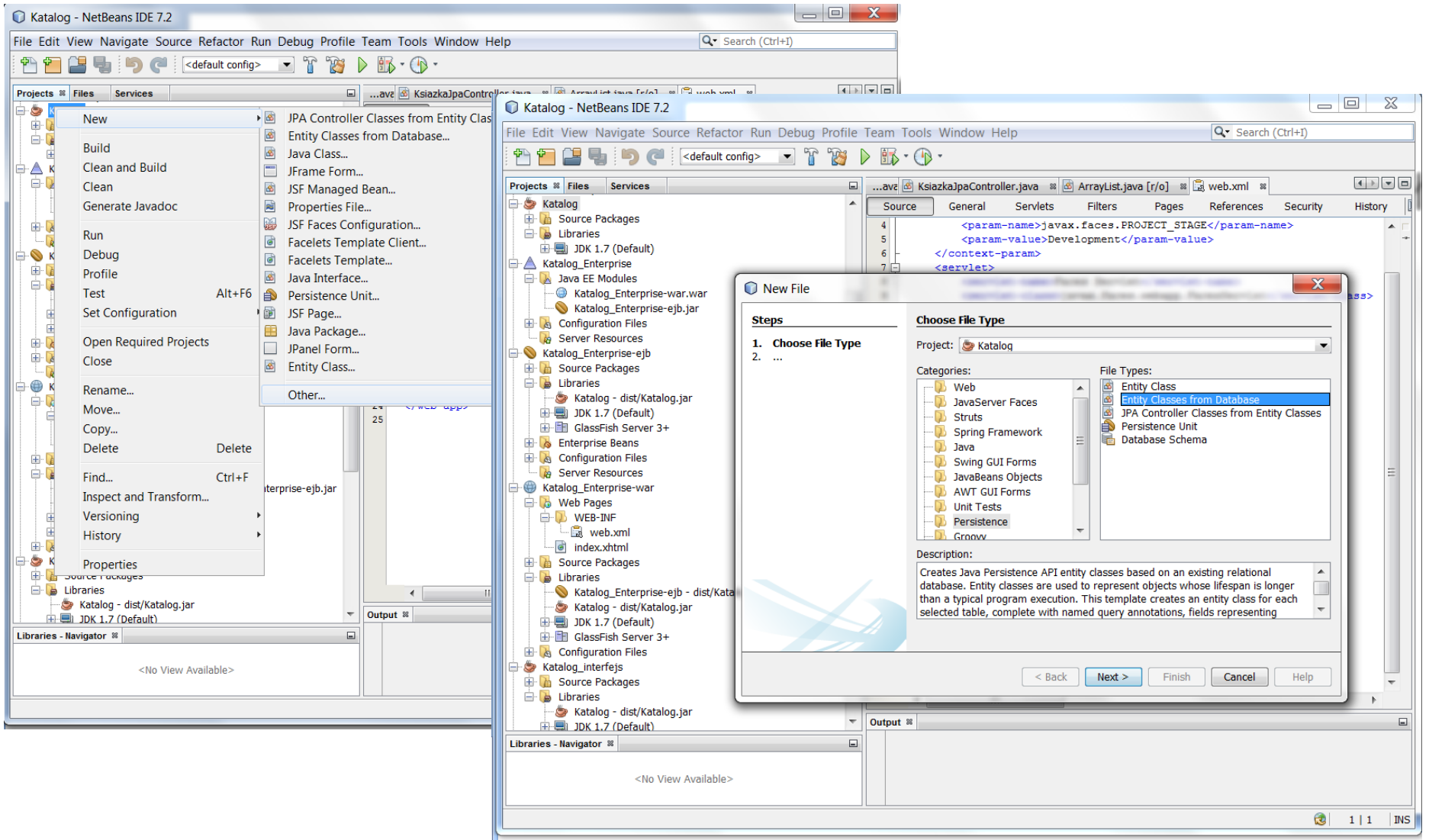




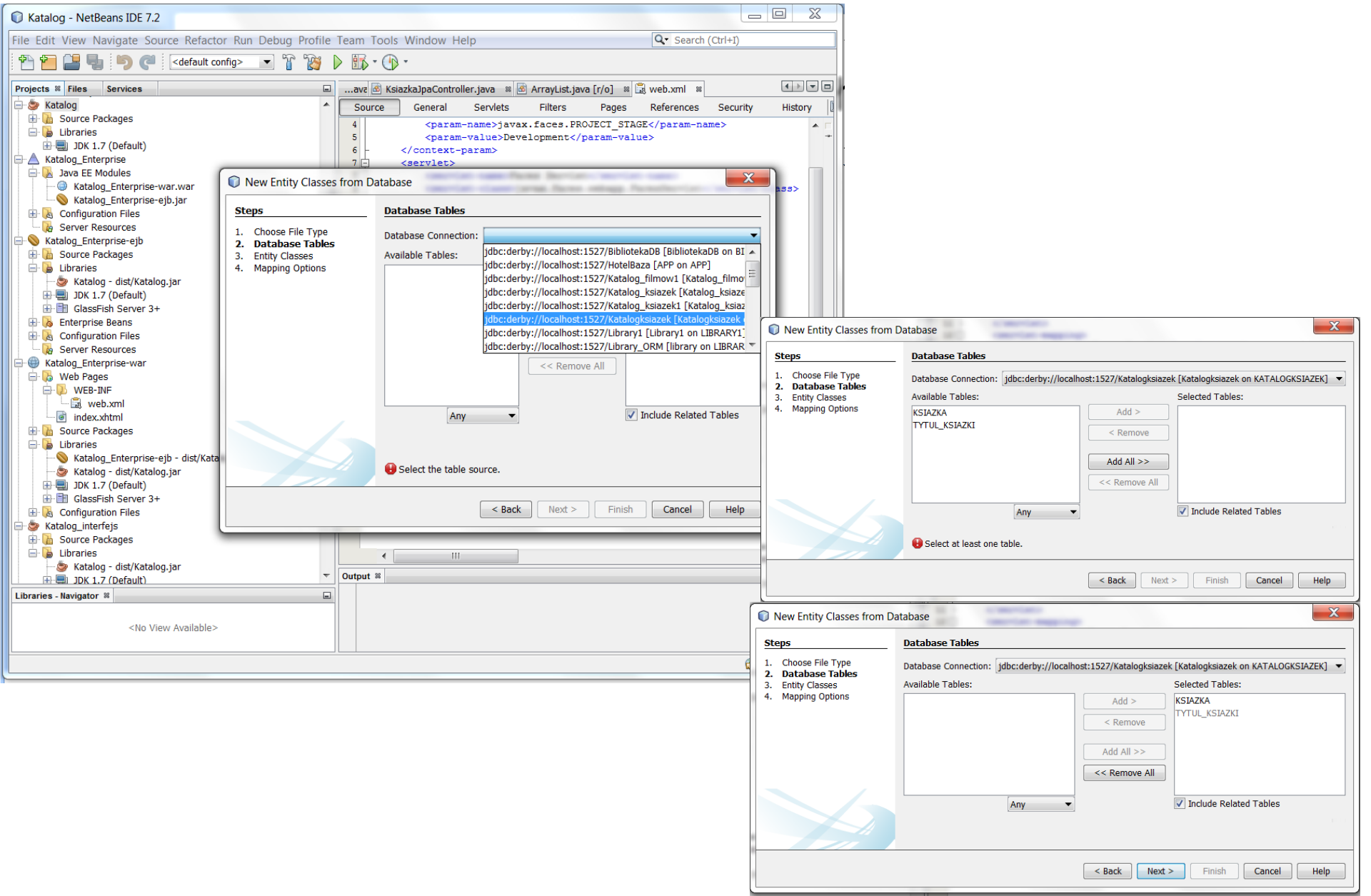
5.2. Widok katalogów Libraries w podanych projektach po zakończeniu dodawania projektu Katalog



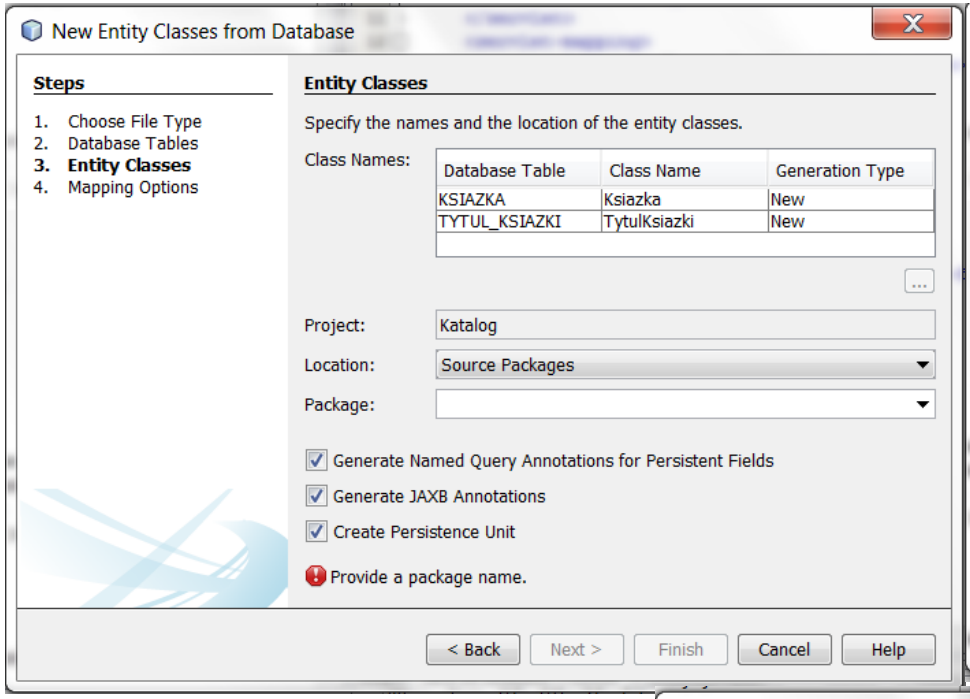
6.1. Dodanie klas typu Entity tworzących obiektowy model danych **warstwy biznesowej** oraz warstwy integracji z bazą danych typu ORM do projektu Katalog, wygenerowanych z bazy danych Katalogksiazek używanej w projekcie internetowym wygenerowanym na podstawie tej bazy danych: należy kliknąć prawym klawiszem na nazwę projektu Katalog w zakładce Projects, wybrać z listy New/Other/Persistence/Entity Classes from Database

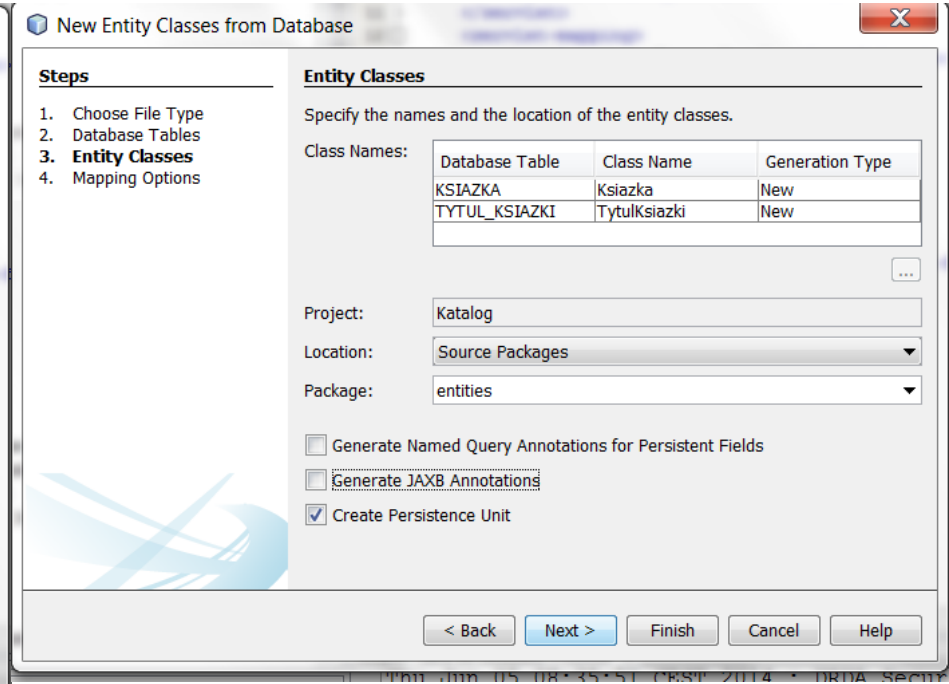


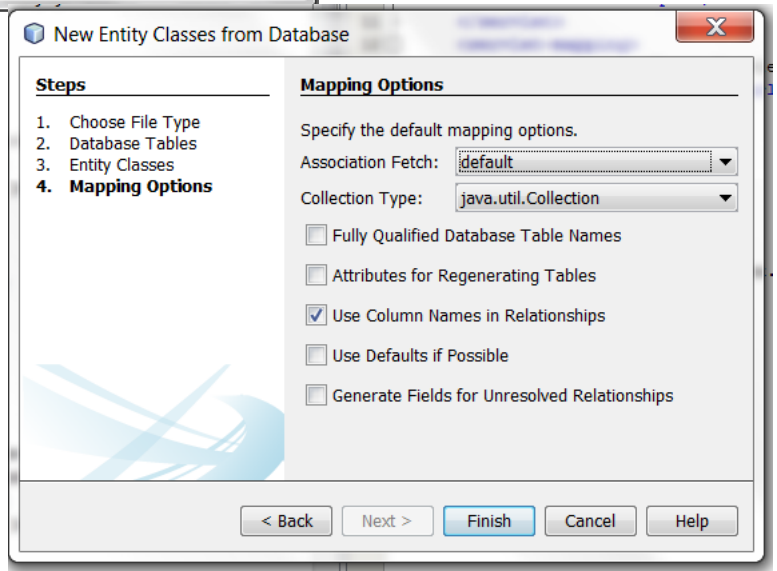
6.2. Wybór z listy bazy danych Katalogksiazek, następnie wybór tabel KSIAZKA oraz TYTUL_KSIAZKI przez Add All>> oraz kliknięcie na klawisz Next



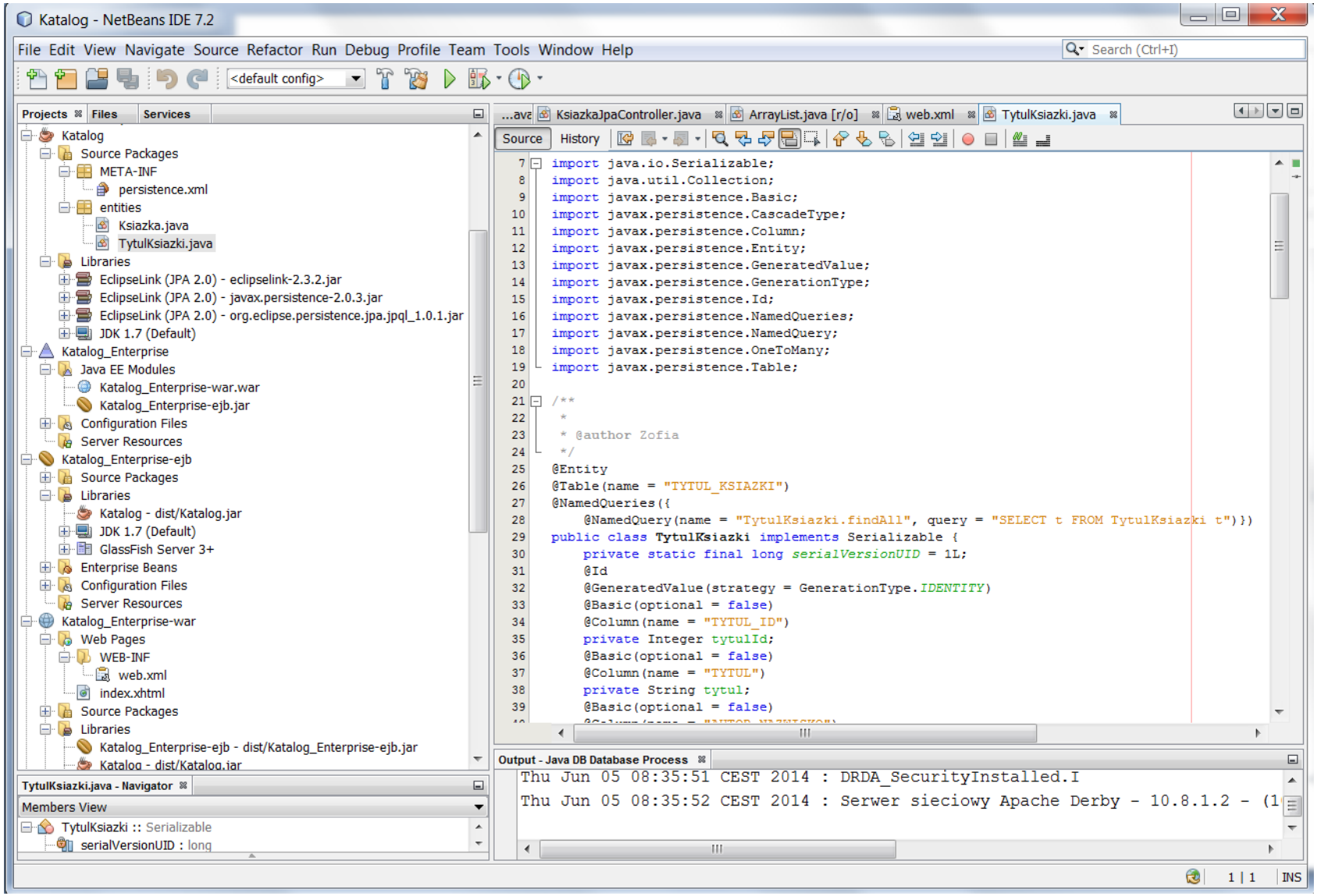
6.3. Podanie nazwy pakietu entities w polu Package, w którym zostaną wygenerowane klasy typu Entity i pozostawienie wybranej opcji Create Persistence Unit. Po naciśnięciu klawisza Next zatwierdzenie procesu klawiszem Finish



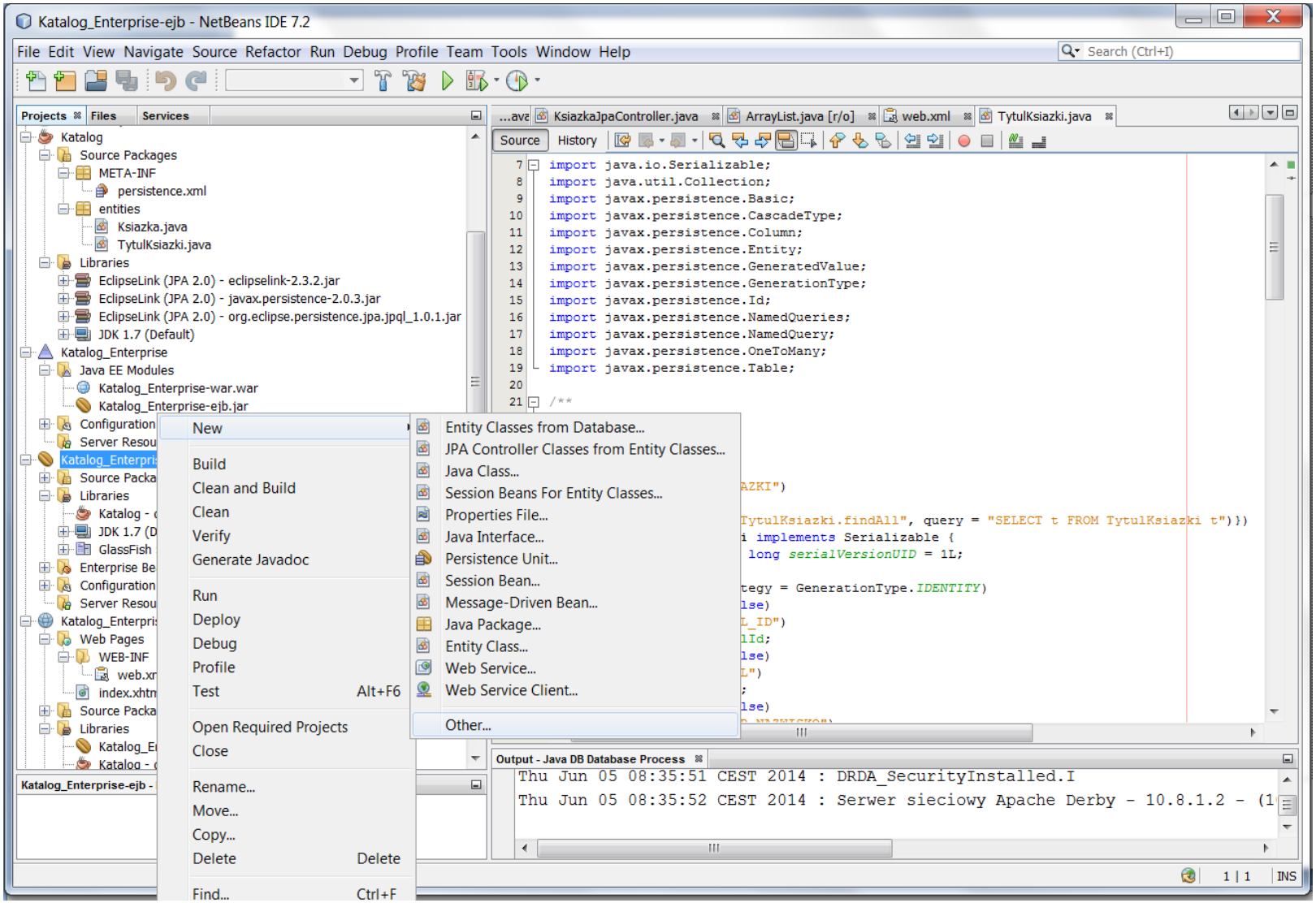




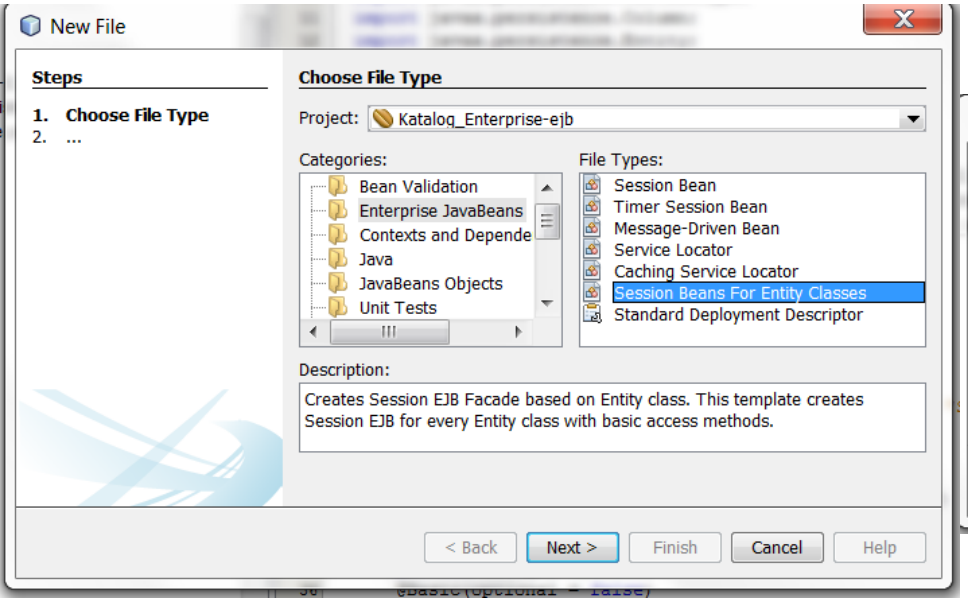
6.4. Wygenerowane klasy w pakiecie entities w projekcie Katalog: TytulKsiazki typu Entity z tabeli TYTUL_KSIAZKI oraz Ksiazka z tabeli KSIAZKA

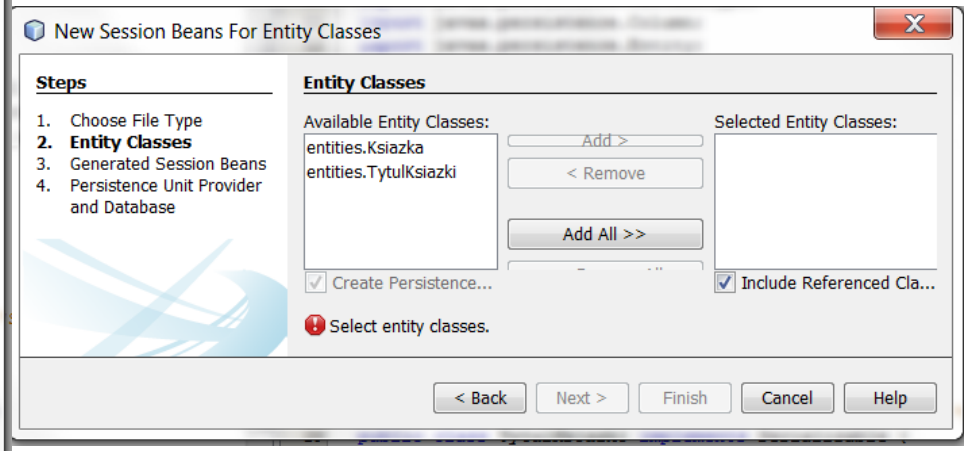


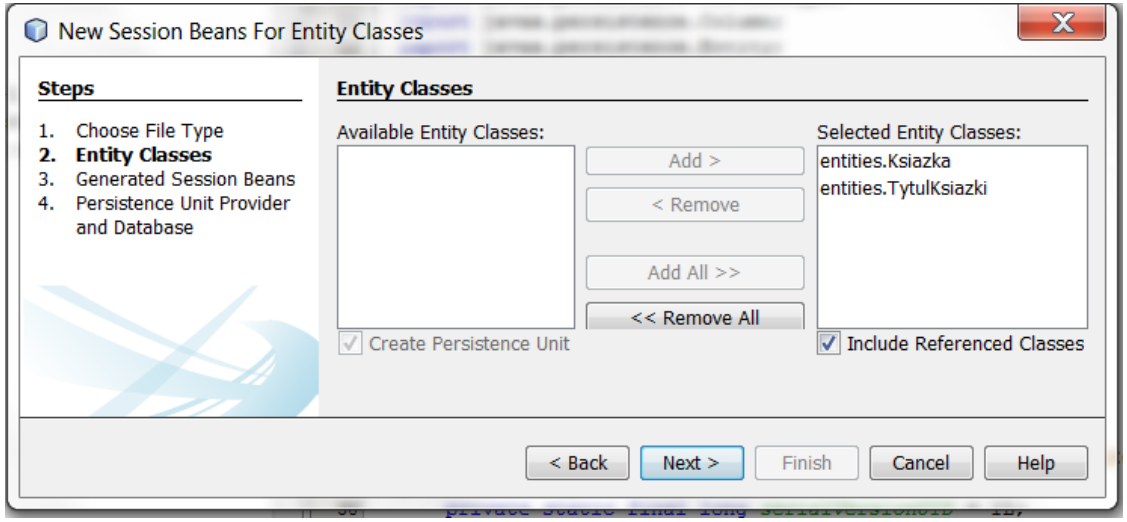
7.1. Wygenerowanie klas typu Session Beans For Entity Classes w module Katalog_Enterprise-ejb, tworzących **warstwę integracji** z bazą danych typu ORM: po kliknięciu prawym klawiszem na nazwę projektu w zakładce Projects wybór pozycji New/Other



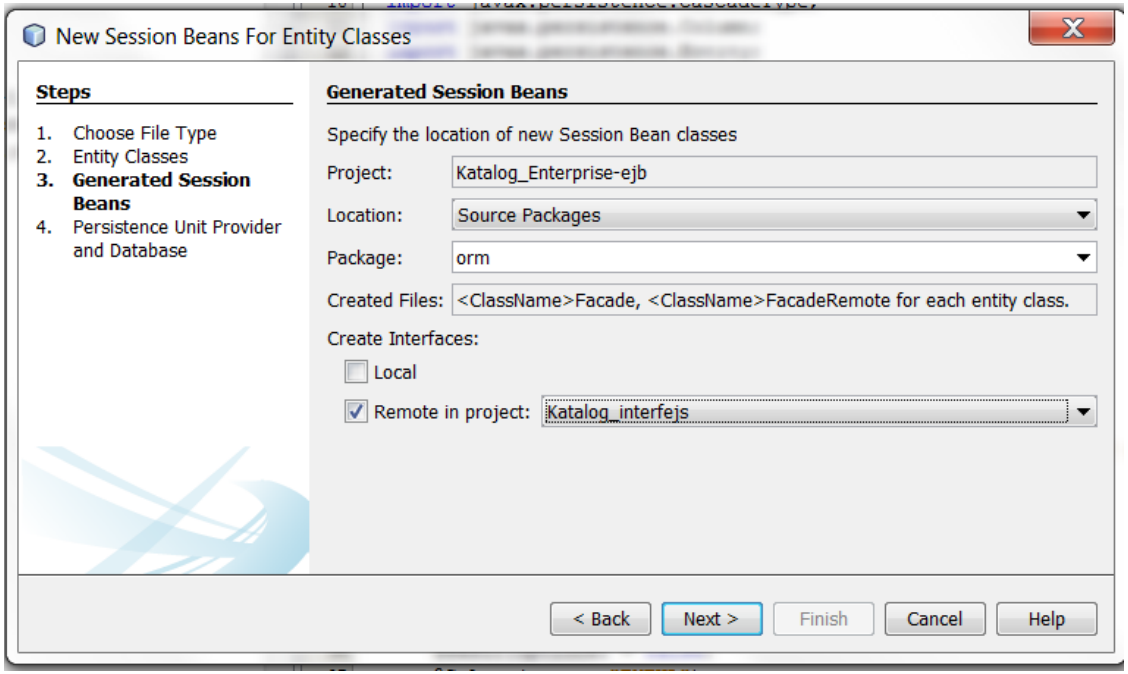
7.2. Następnie wybór Enterprise JavaBeans/Session Beans For Entity Classes, wybór klas typu Entity: entites.Ksiazka oraz entities.TytulKsiazki klawiszem Add All i naciśnięcie klawisza Next

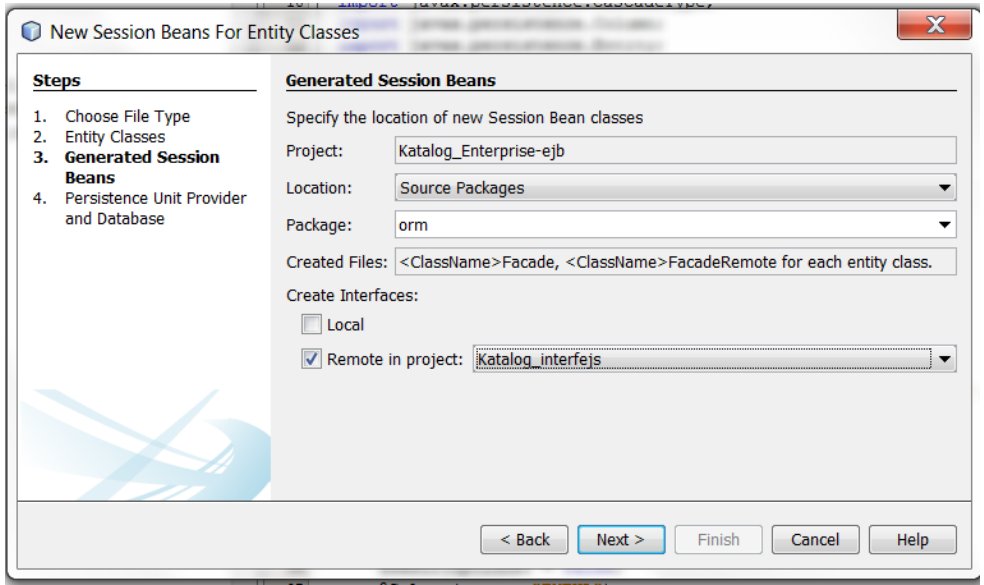






7.3. Utworzenie pakietu orm, zaznaczenie opcji Remote in project, wybór z listy projektu Katalog_interfejs i naciśnięcie klawisza Next

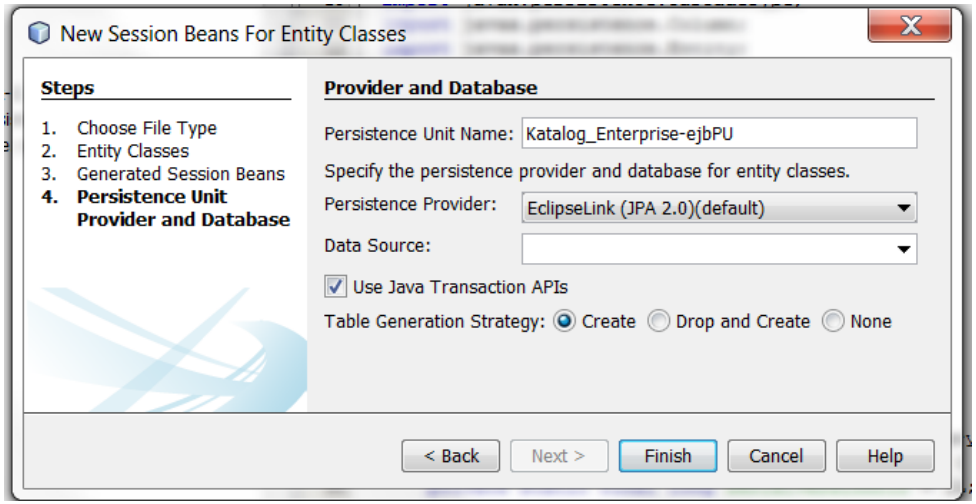


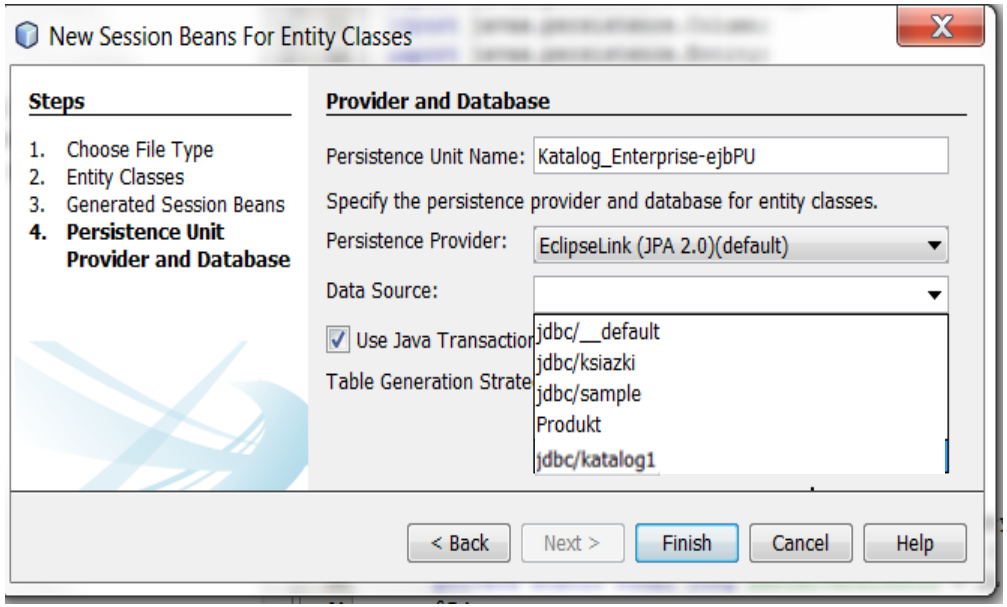


7.4. Rozwinięcie listy Data Source i wybór pozycji **jdbc/katalog1**, łączący z bazą danych **jdbc:derby://localhost:1527/Katalogksiazek [Katalogksiazek on KATALOGKSIAZEK]**

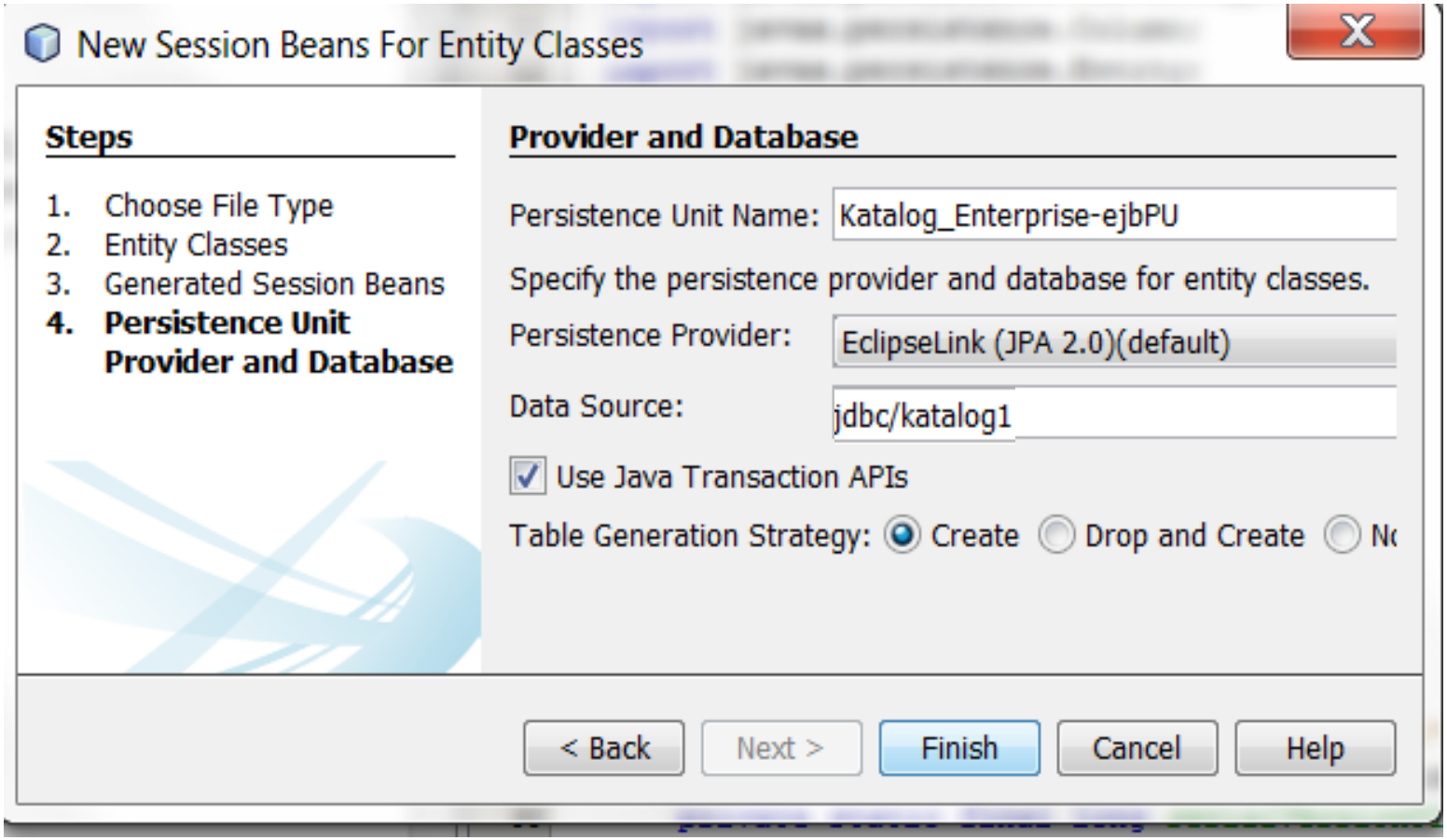
(wykonany wg slajdów 27-33 w pliku

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/PiWSI/Aplikacja_internetowa_JSF_z_bazy_danych_2015.pdf)

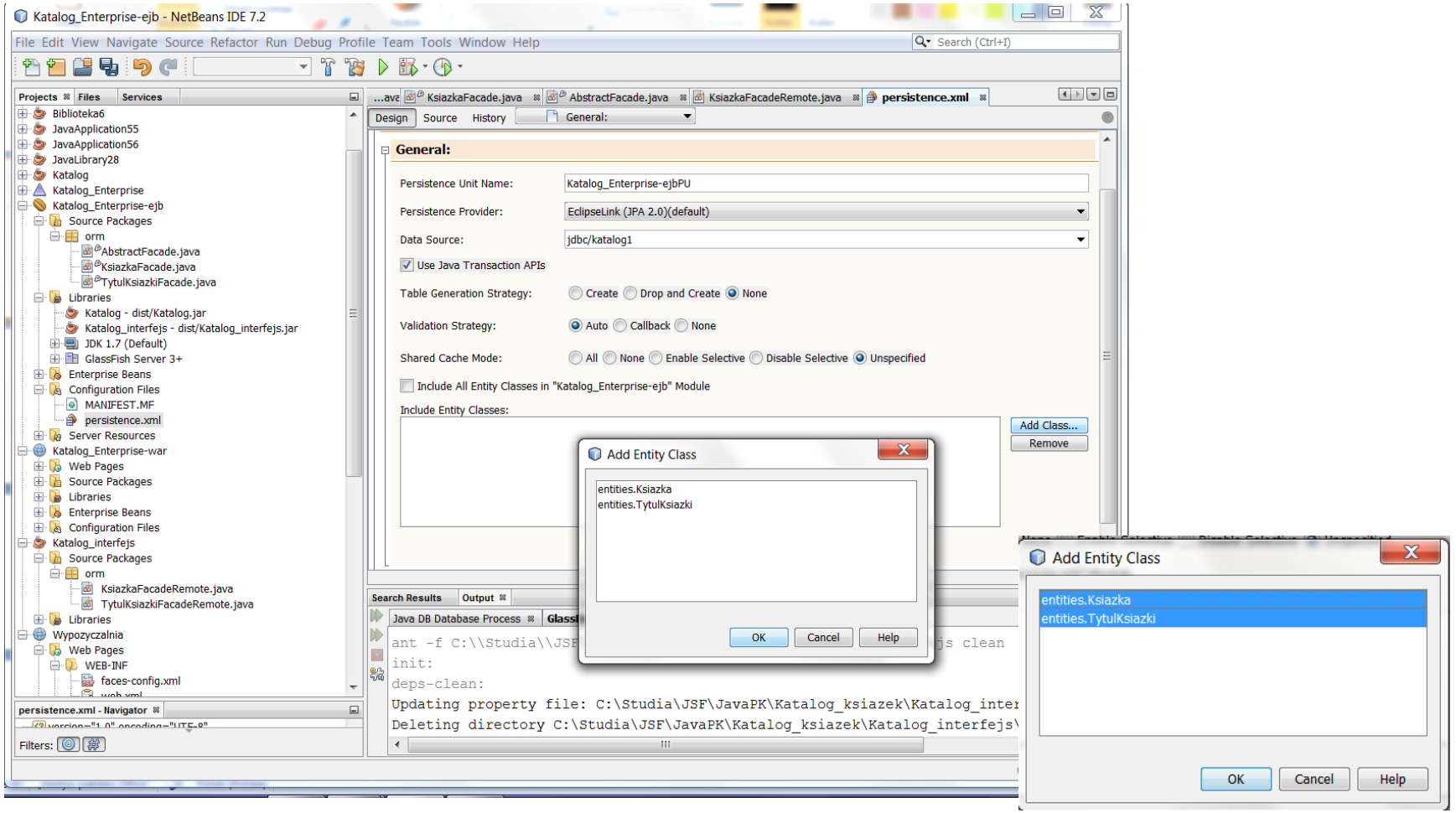




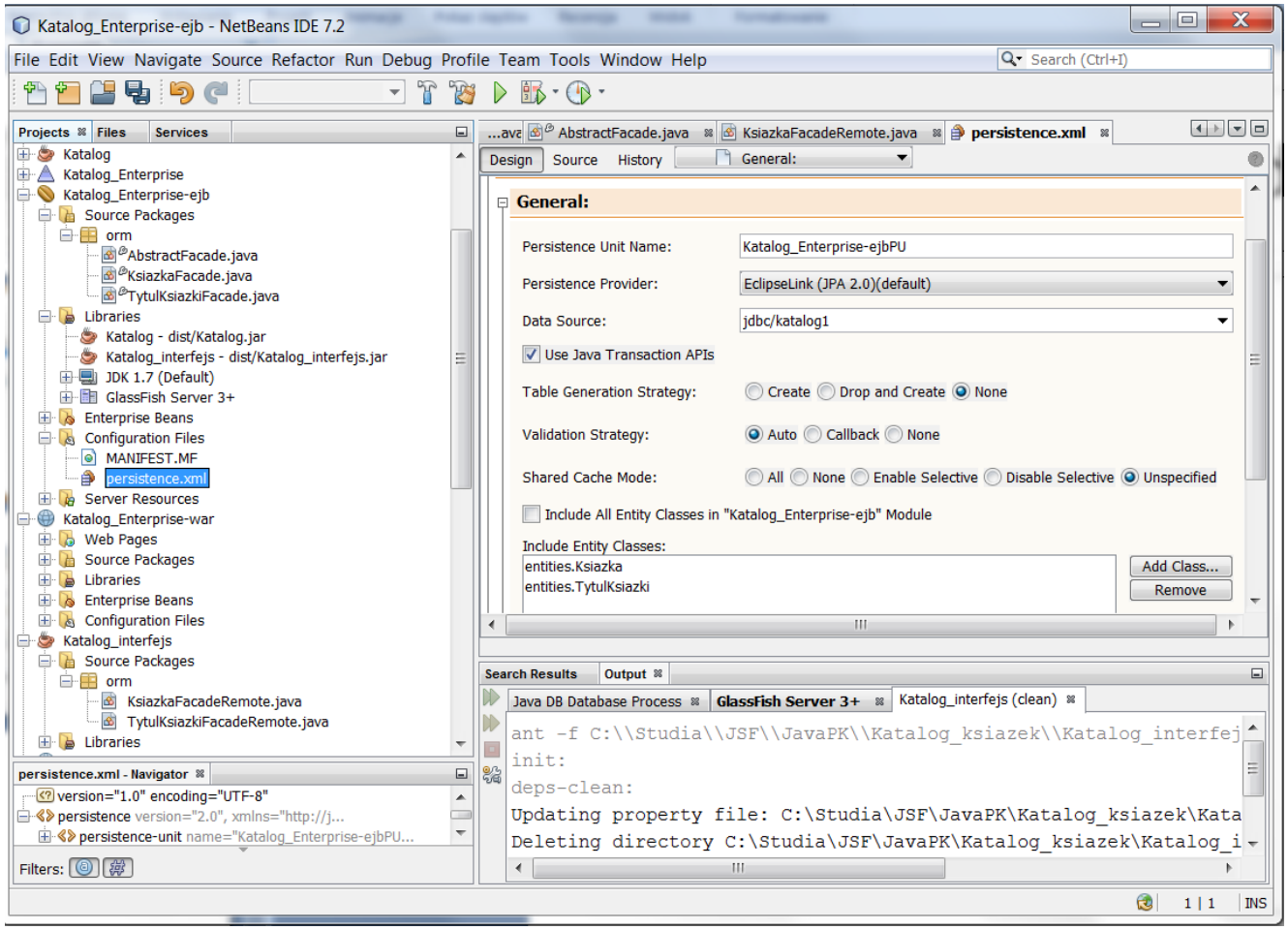
7.5. Zatwierdzenie czynności klawiszem Finish.



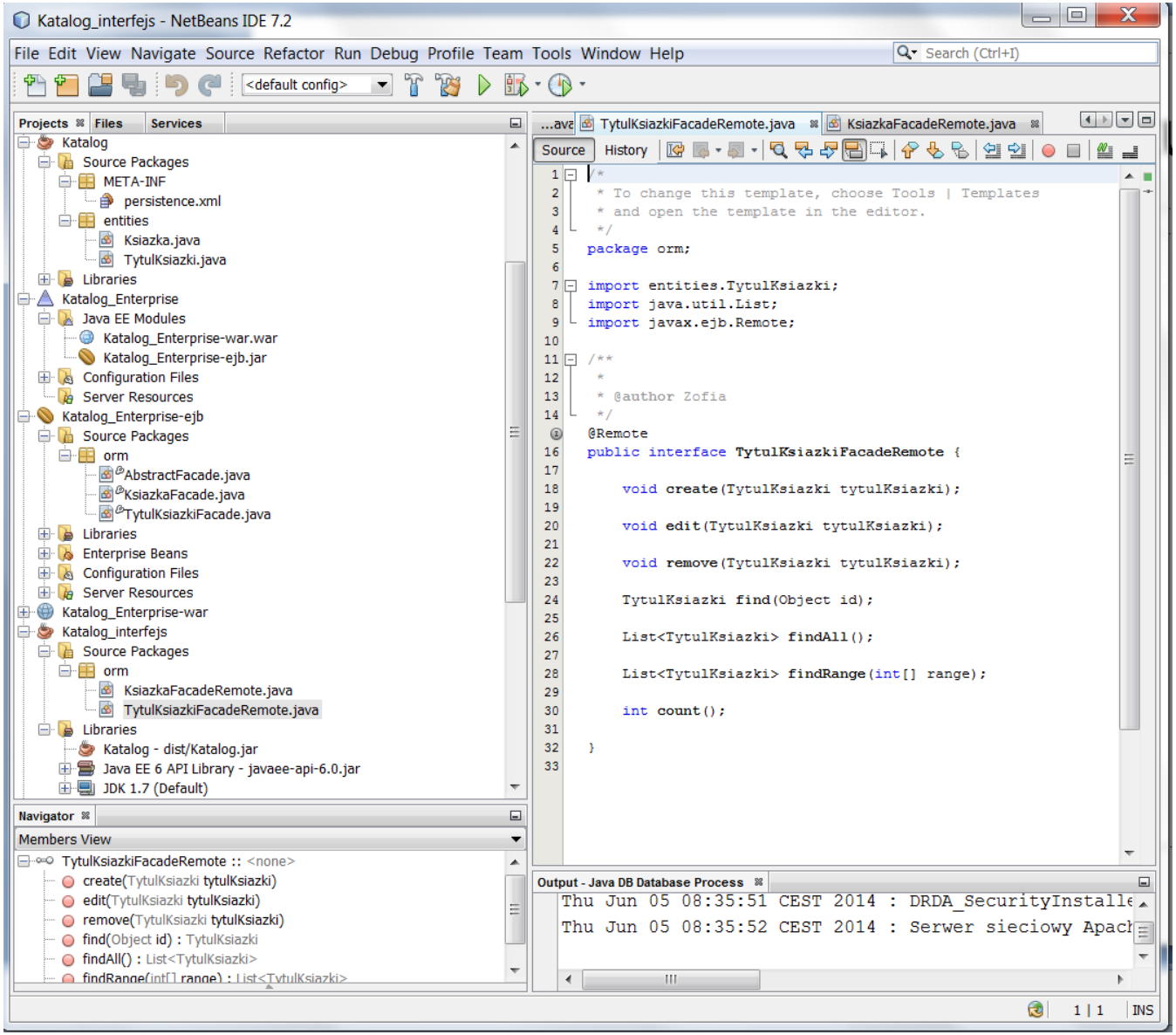
7.6. Ustawienie w module Persistence Unit Table Generation Strategy na None, usunięcie opcji Include All Entities Classes in „Katalog_Enterprise-ejb” Module oraz wybór klas z pakietu entities klawiszem Add Class…. Po zaznaczeniu tych klas zatwierdzenie operacji klawiszem OK.



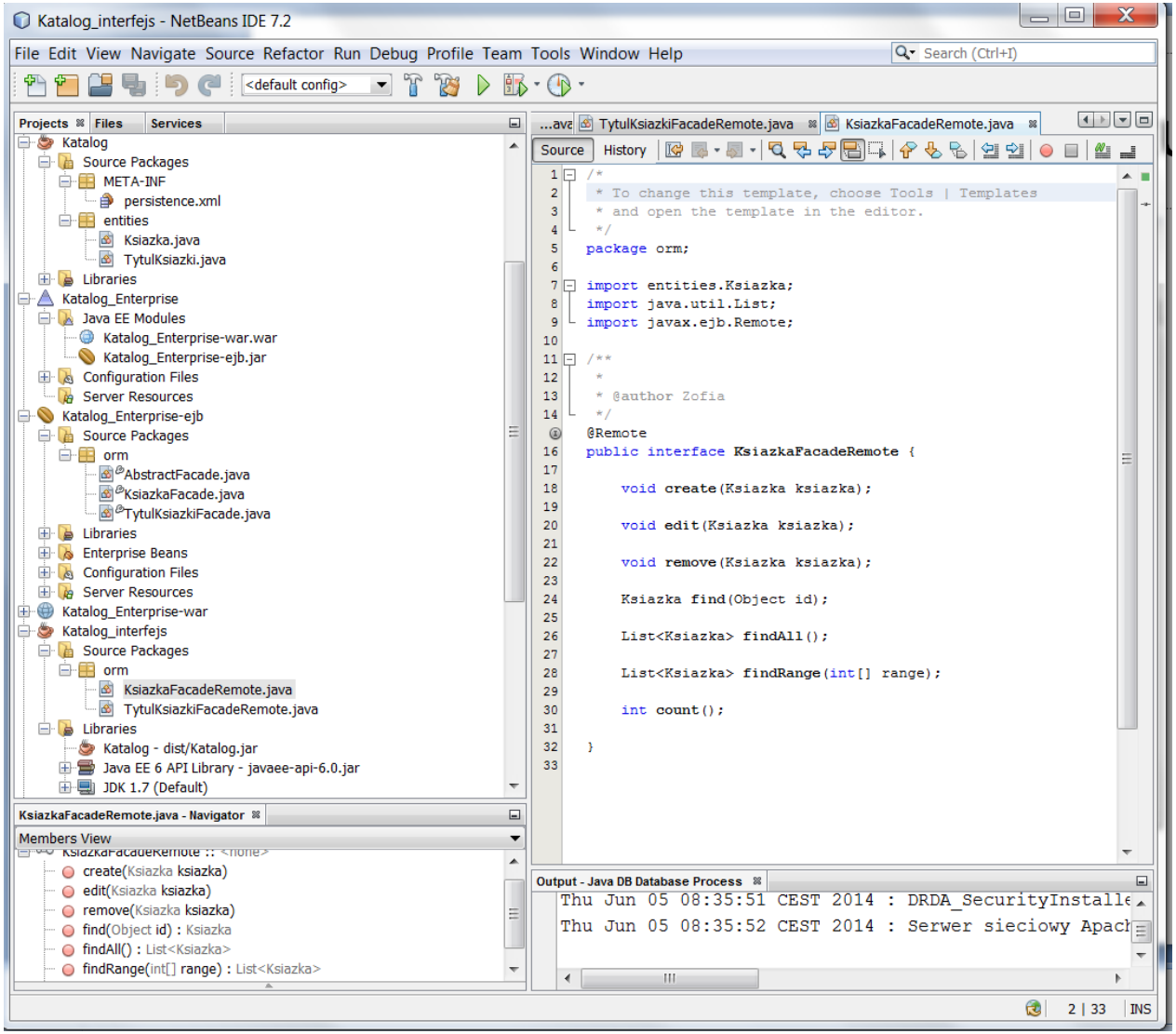
7.7. Zakończenie procesu generowania klas należących do warstwy integracji danych za pomocą technologii ORM.



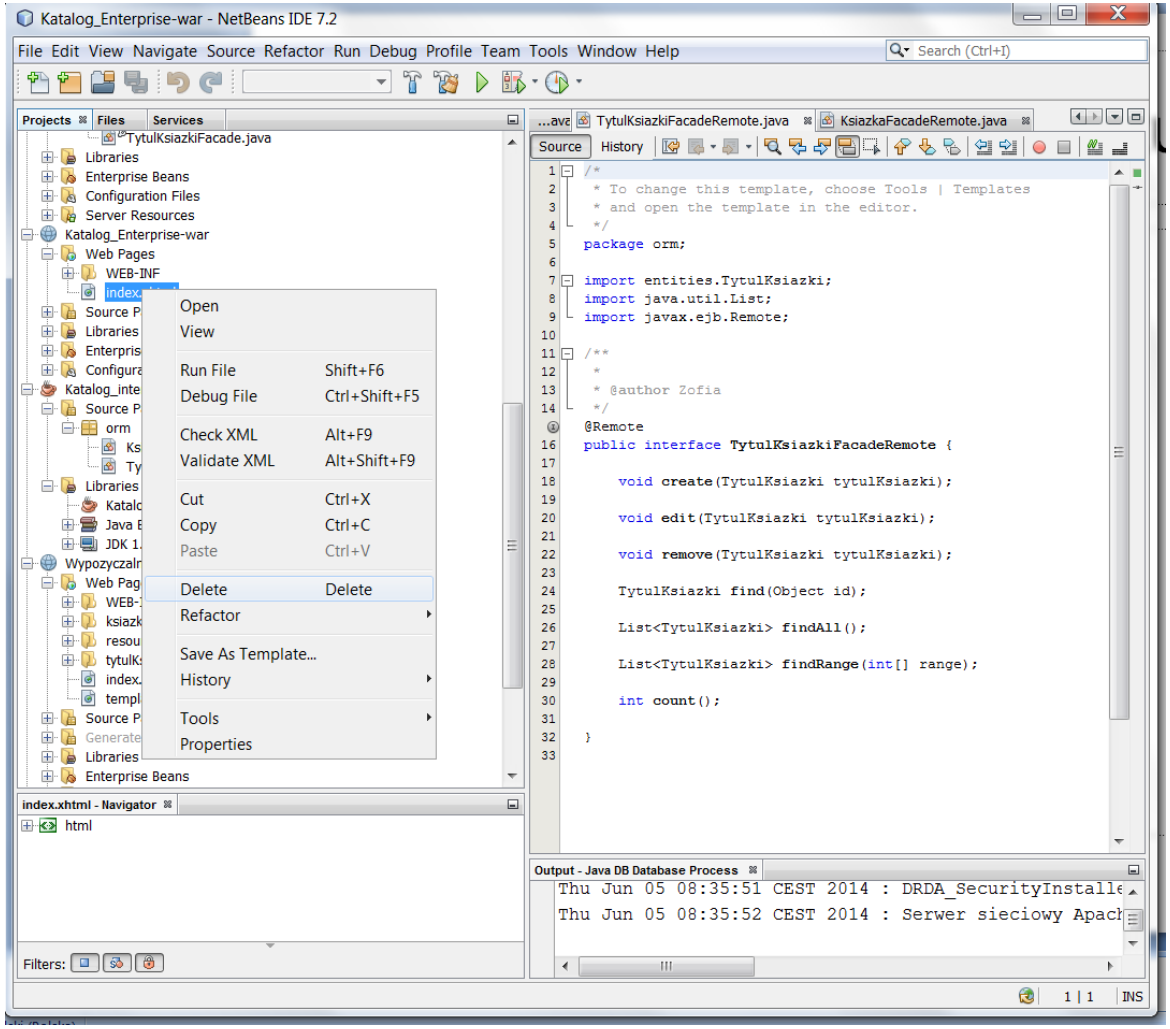
7.8. Interfejs TytulKsiazkiFacadeRemote komponentu TytulKsiazkiFacade typu Session Bean for Entity class w projekcie Katalog_interfejs.



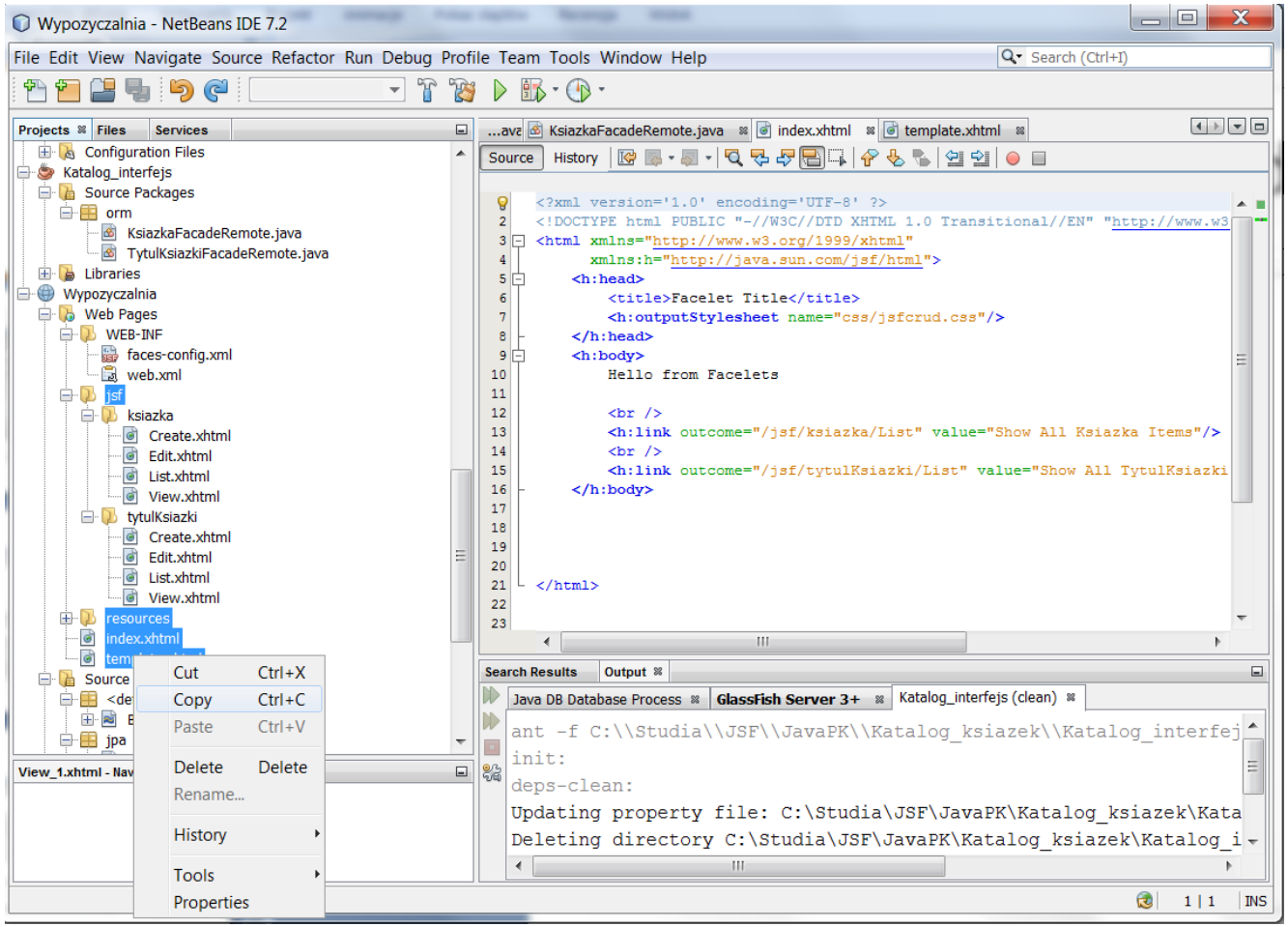
7.9. Interfejs KsiazkaFacadeRemote komponentu KsiazkaFacade typu Session Bean for Entity class w projekcie Katalog_interfejs.



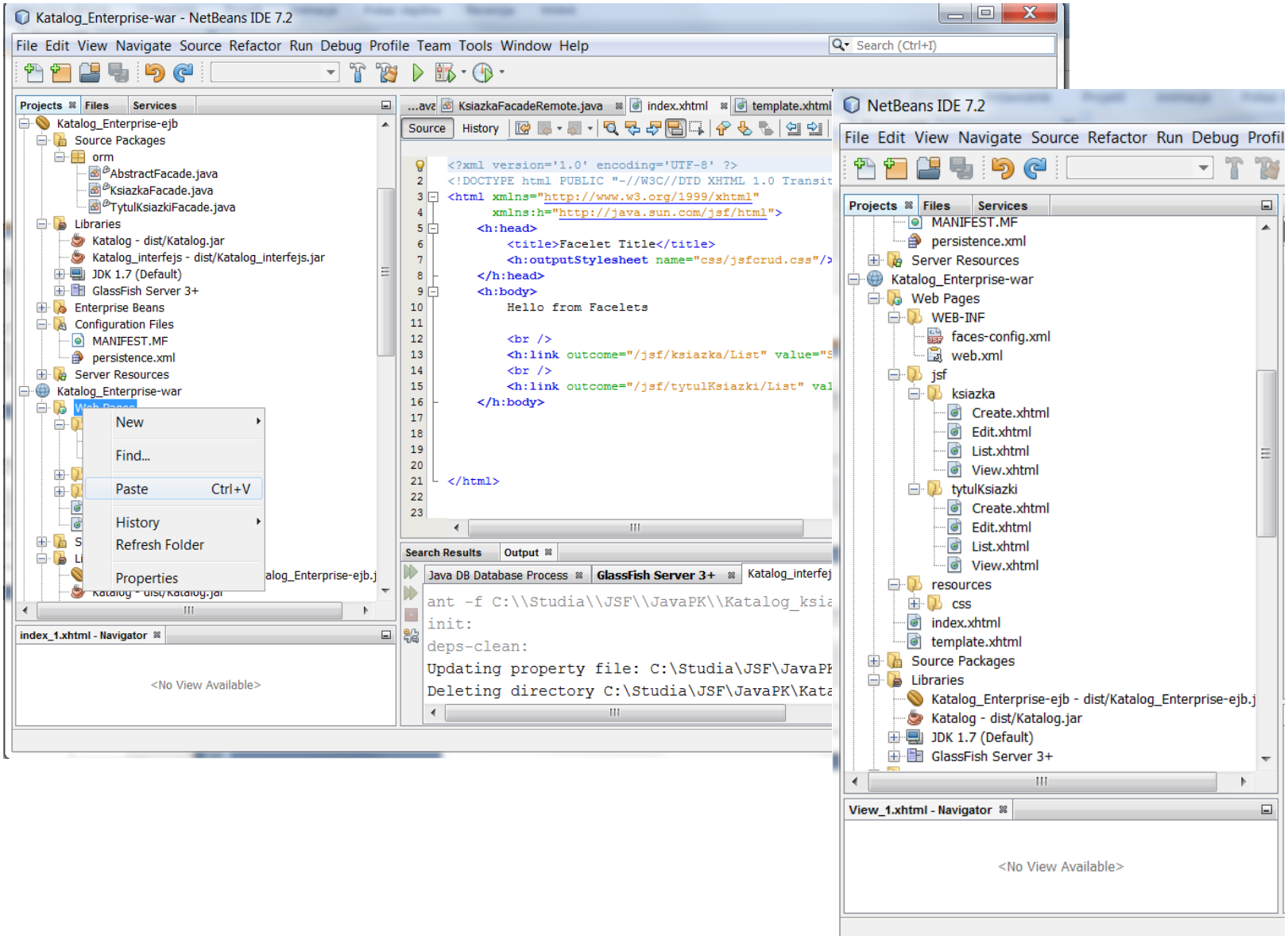
8.1. Przebieg wykonania kopii warstwy prezentacji w module Katalog_Enterprise-war z projektu Wypozyczalnia typu Java Web, opartego na technologii JavaServer Faces 2.1, wygenerowanego na podstawie bazy danych Katalogksiazek – na początku usunięcie pliku index.xhtml w module Katalog_Enterprise-war



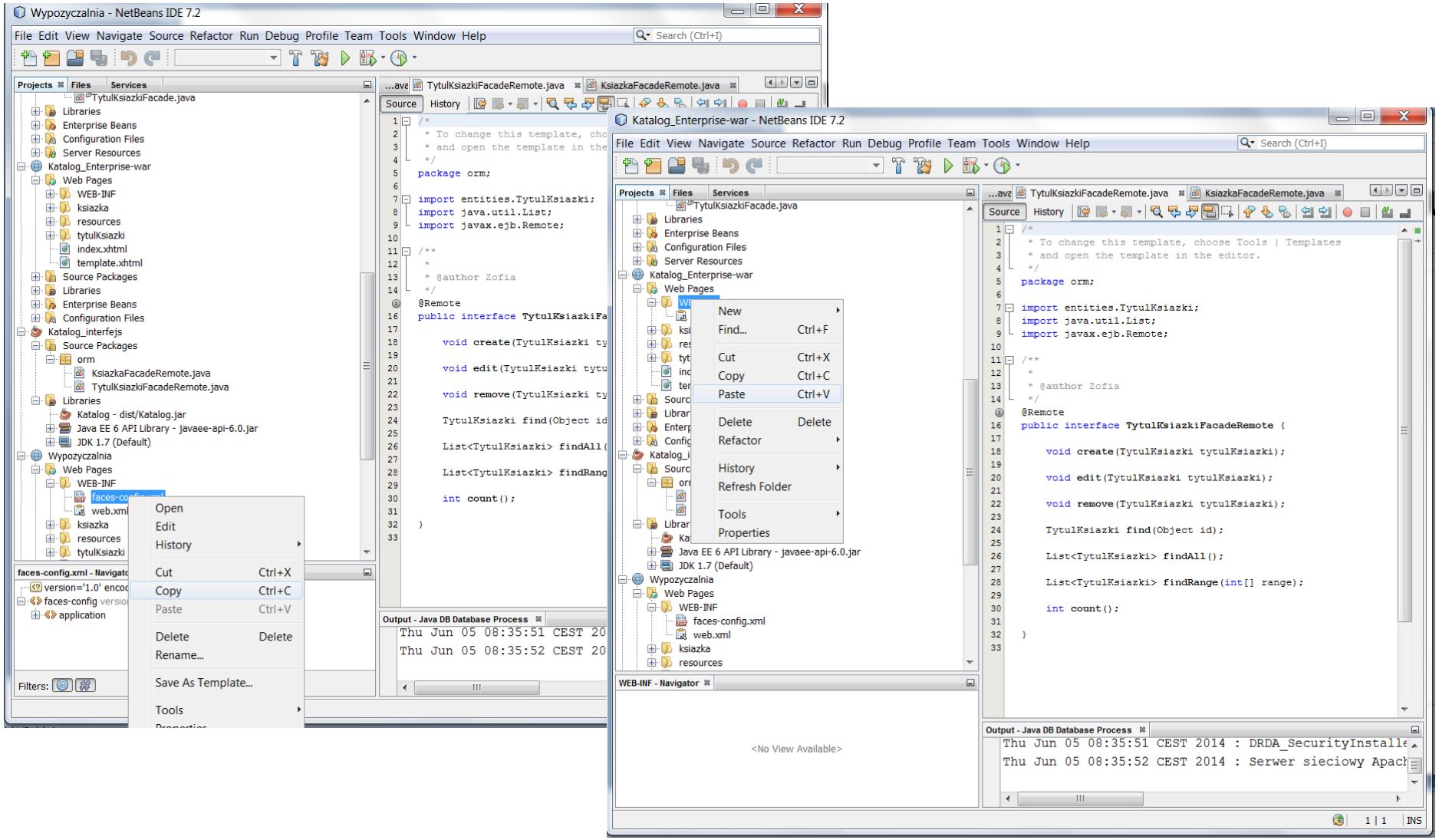
8.2. Wykonanie kopii katalogów i plików z katalogu Web Pages w projekcie Wypozyczalnia : katalogu jsf, zawierającego strony xhtml, katalogu resources z arkuszami stylów typu css, plik index.xhtml oraz szablon stron template.xhml.



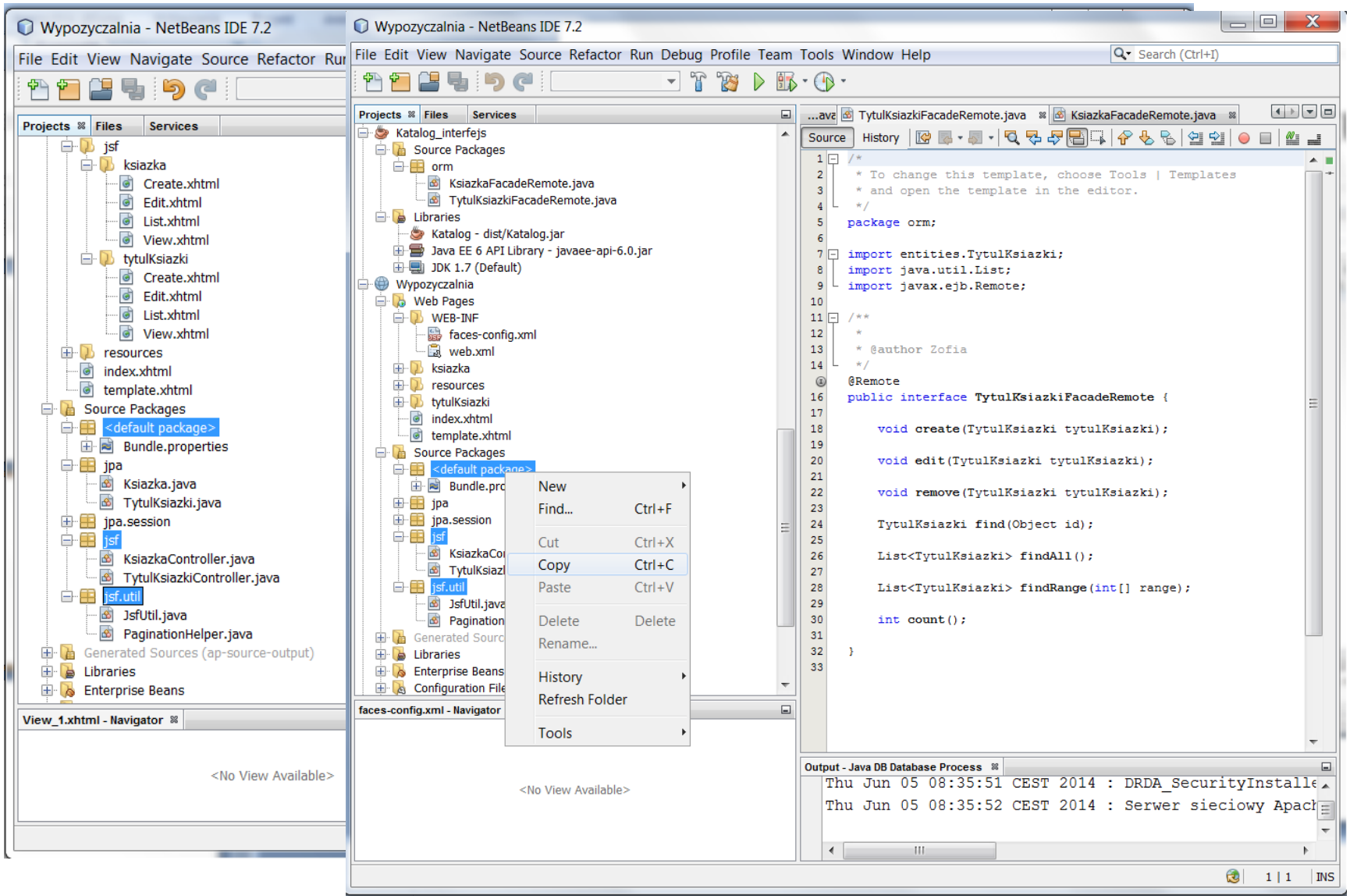
8.3. Wklejenie katalogów i plików podanych w p. 8.3 do katalogu Web Pages w module Katalog_Enterprise-war



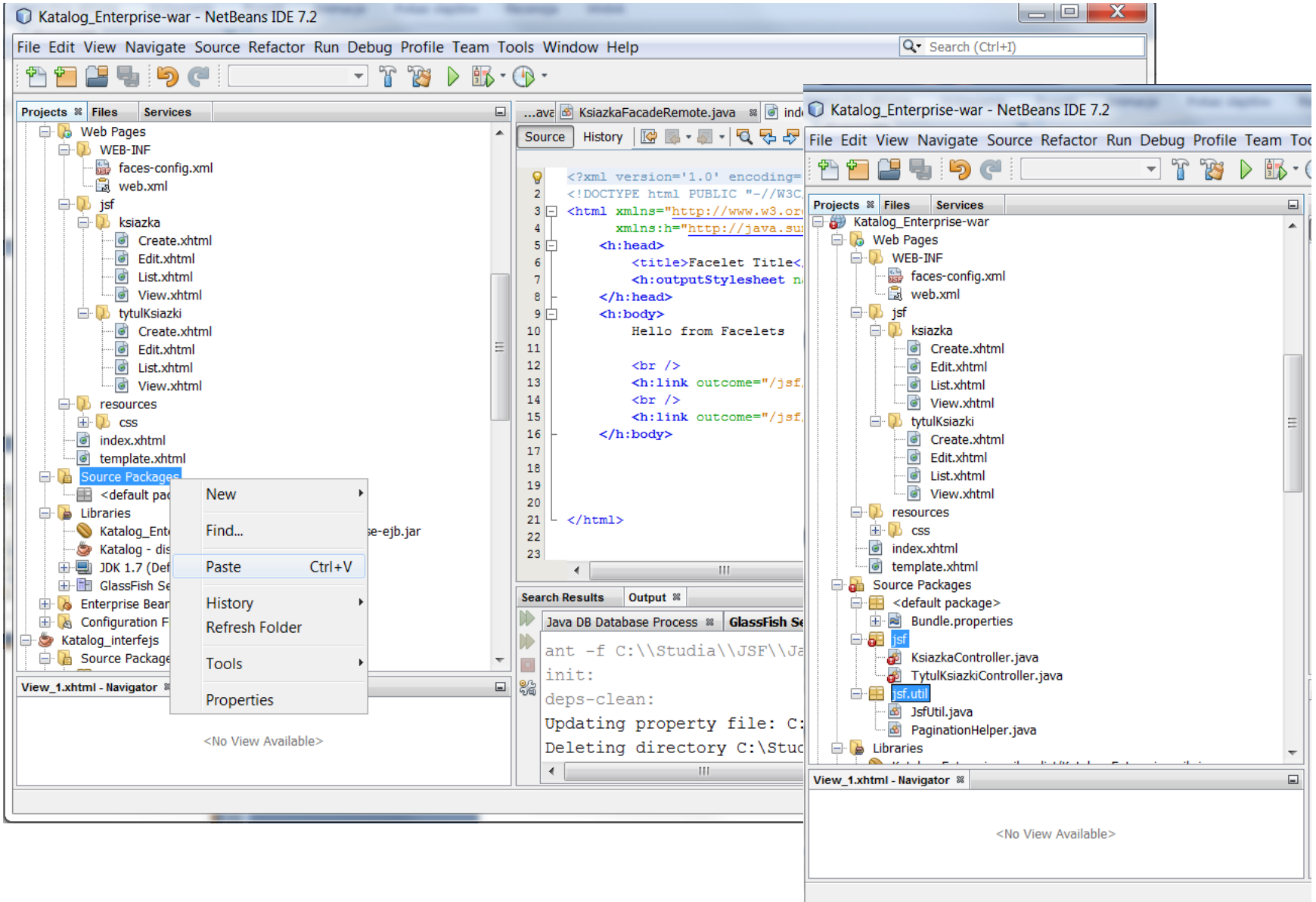
8.4. Skopiowanie pliku faces-config.xml, znajdującego się w podkatalogu WEB-INF katalogu Web Pages (z projektu Wypozyczalnia do modułu Katalog_Enterprise-war).



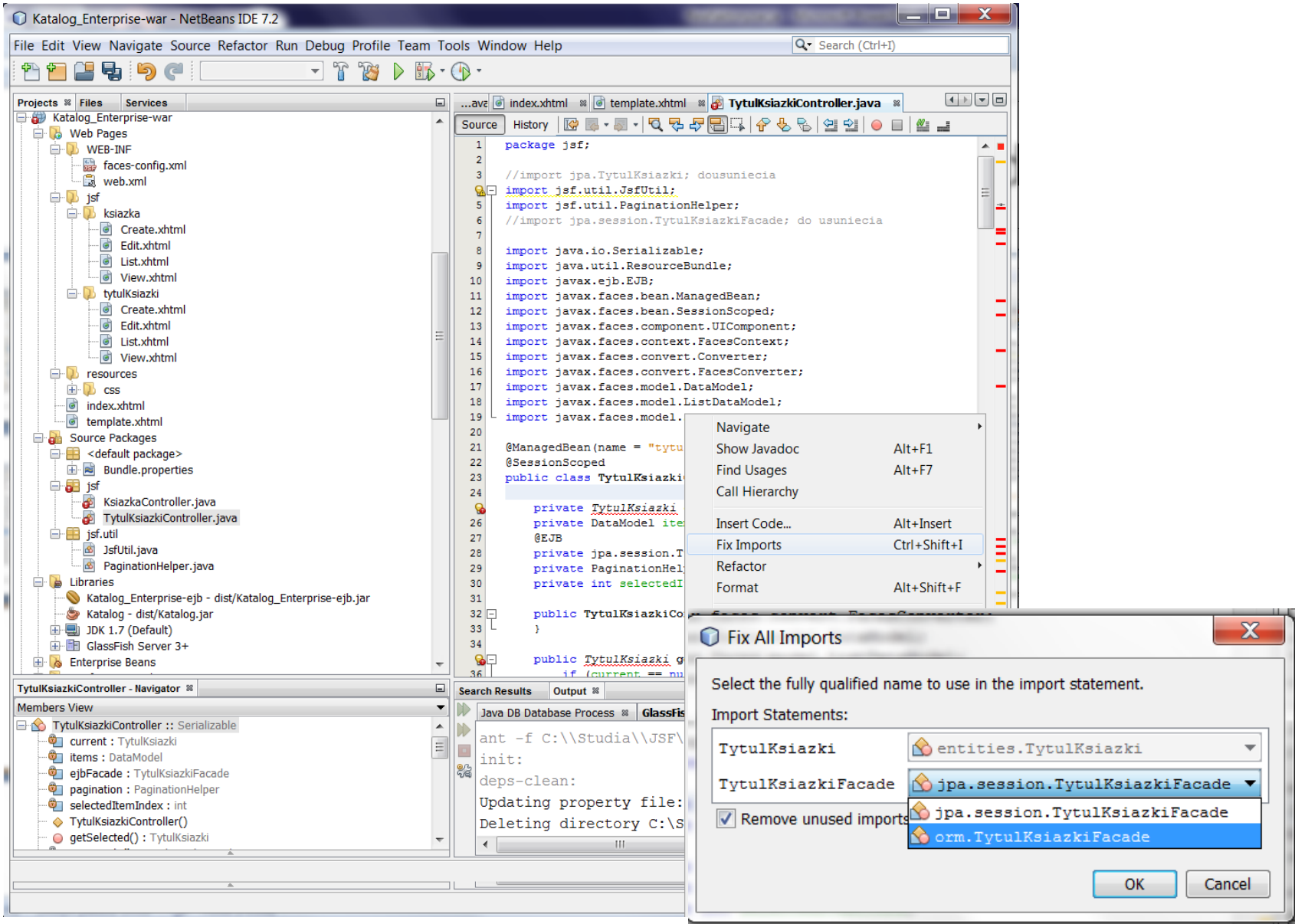
8.5. Skopiowanie następujących pakietów z katalogu Source Packages w projekcie Wypozyczalnia : default package, gdzie znajduje się plik Bundle.properties, pakiet jsf, zawierający komponenty typu Managed Bean oraz pakiet jsf.util, zawierający klasy narzędziowe stronicowania komponentu DataTable oraz obsługi błędów



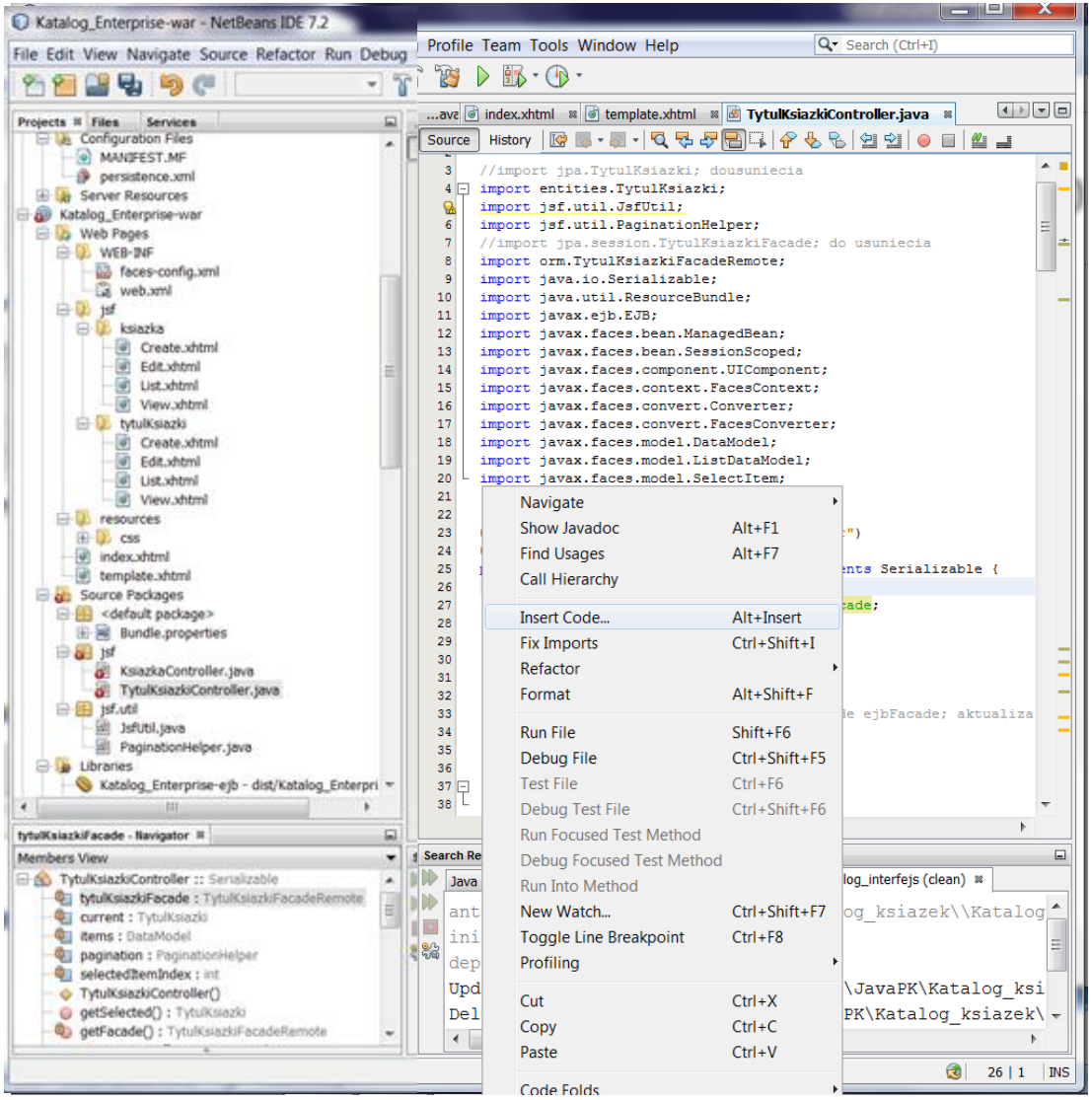
8.6. Wklejenie katalogów i plików podanych w p. 8.5 do katalogu Source Packages w module Katalog_Enterprise-war

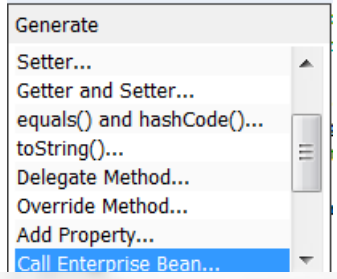


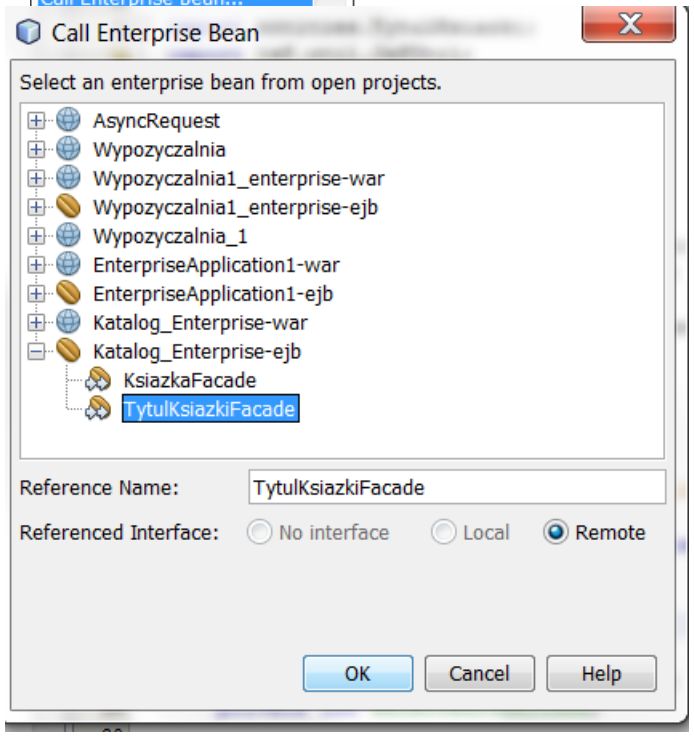
8.7. Usuwanie błędów kompilacji w klasie TytulKsiazkiController - po kliknięciu prawym klawiszem na klasę TytulKsiazkiController należy wybrać pozycję Fix Imports i wybrać właściwe klasy w okienku Fix All Imports.



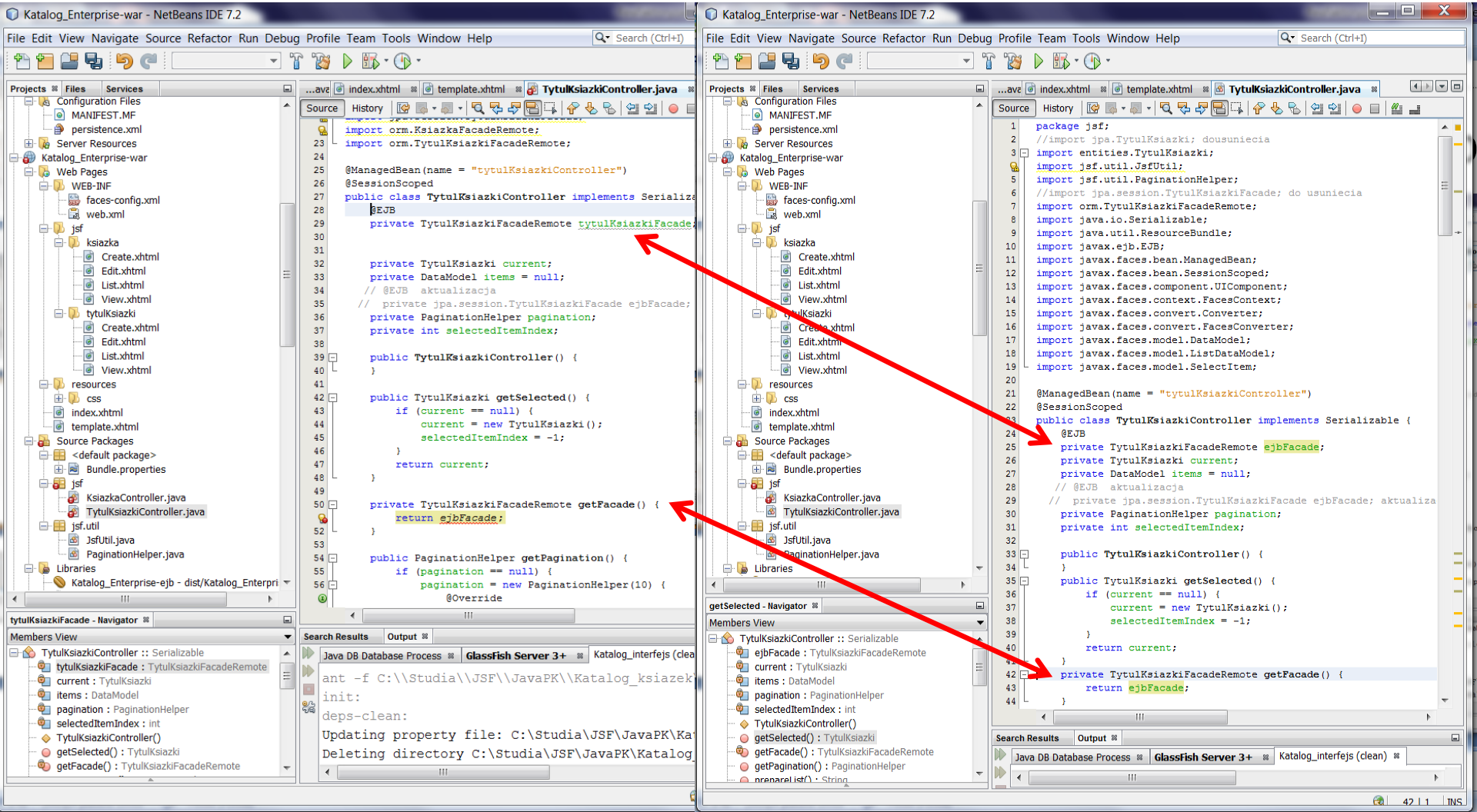
8.8. Dodanie dostępu do komponentów z warstwy integracji - po kliknięciu prawym klawiszem na klasę TytulKsiazkiController należy wybrać pozycję Insert Code…, wybrać pozycję Call Enterprise Bean… w okienku Generate. Następnie, należy wybrać komponent typu TytulKsiazkiFacade w oknie Select an enterprise bean from open projects.



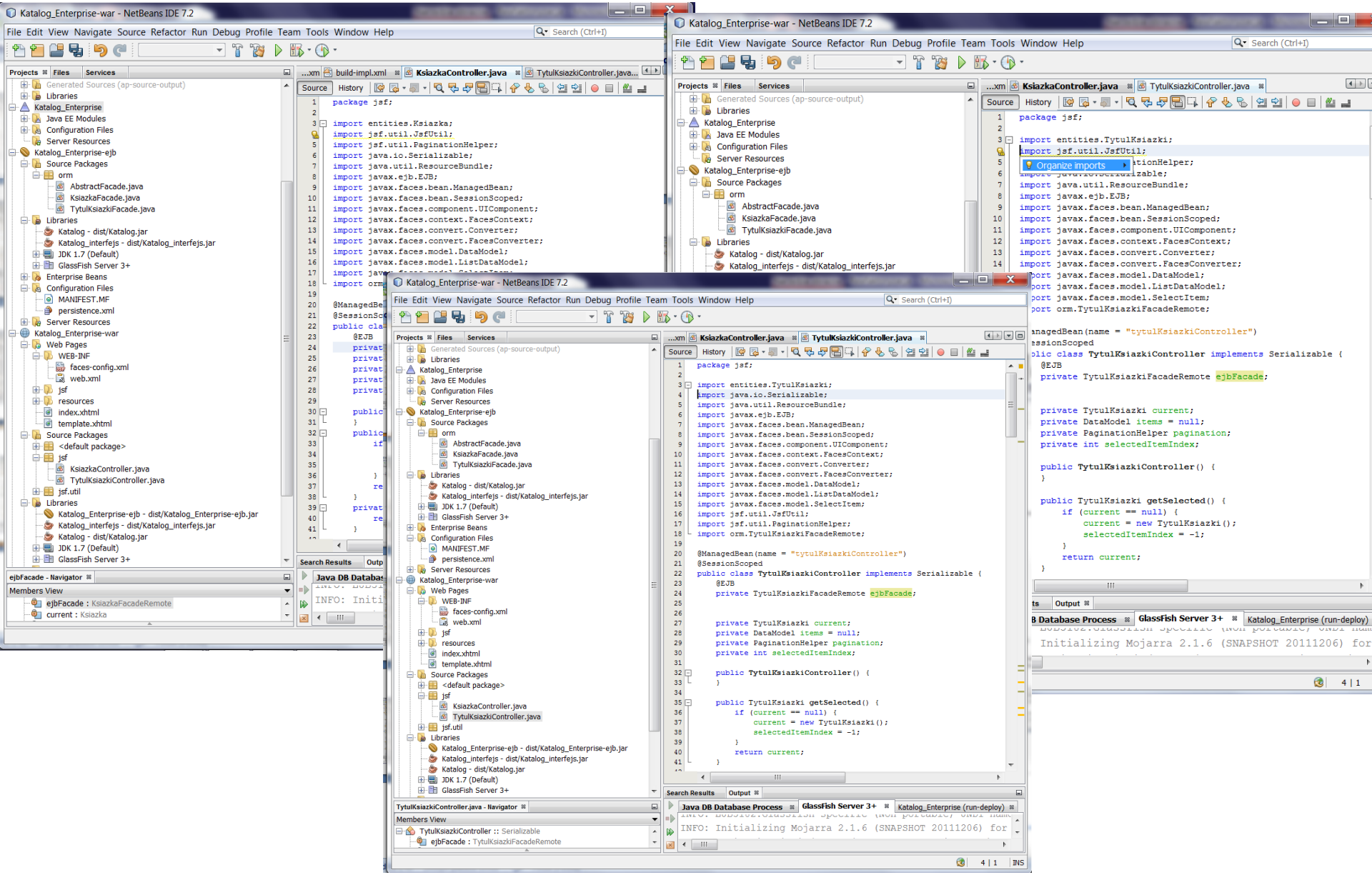




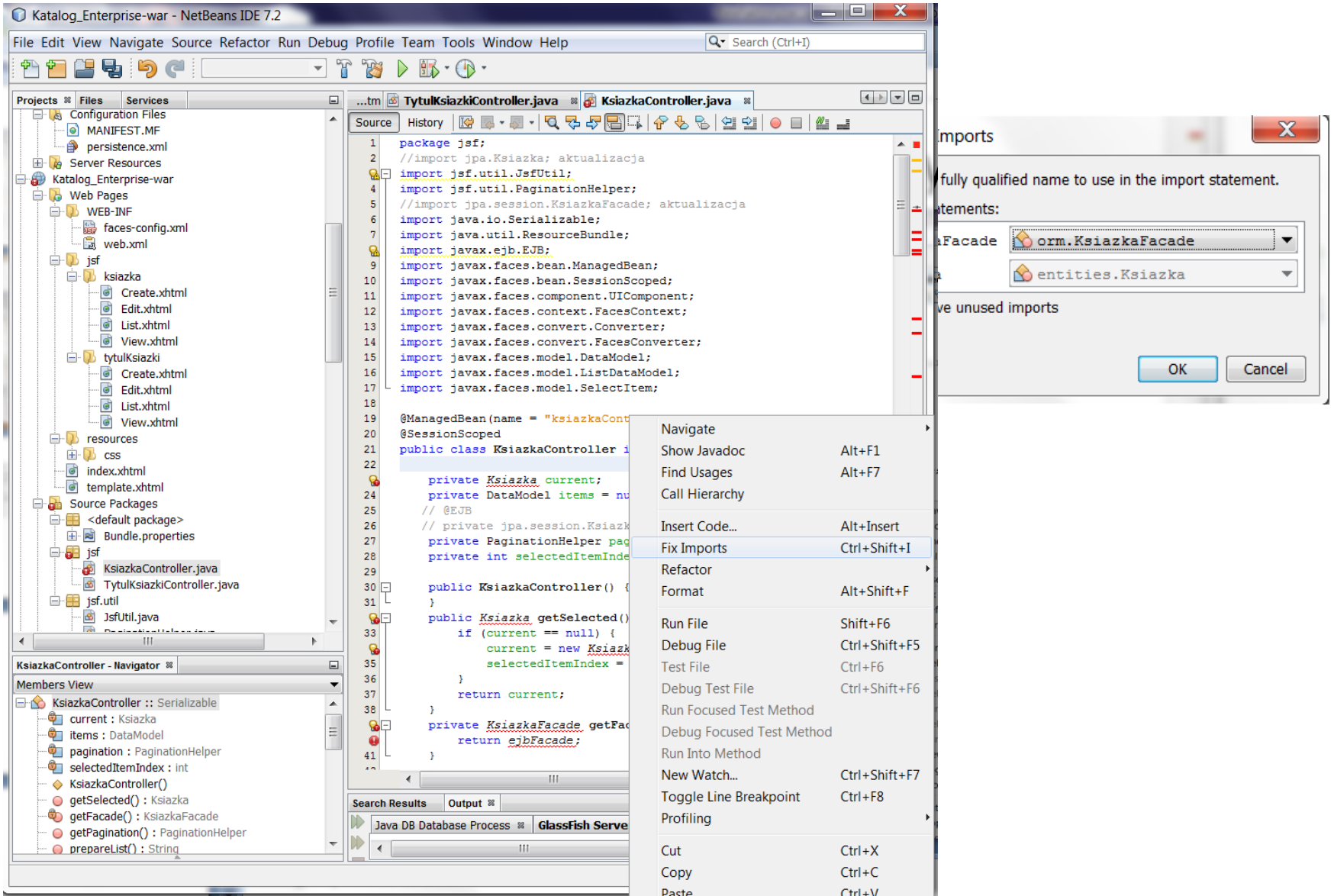
8.9. Poprawa kodu w klasie TytulKsiazkiController wynikająca ze zmiany nazwy dodanego obiektu oraz zmiany typu tego obiektu dodanego za pomocą adnotacji.



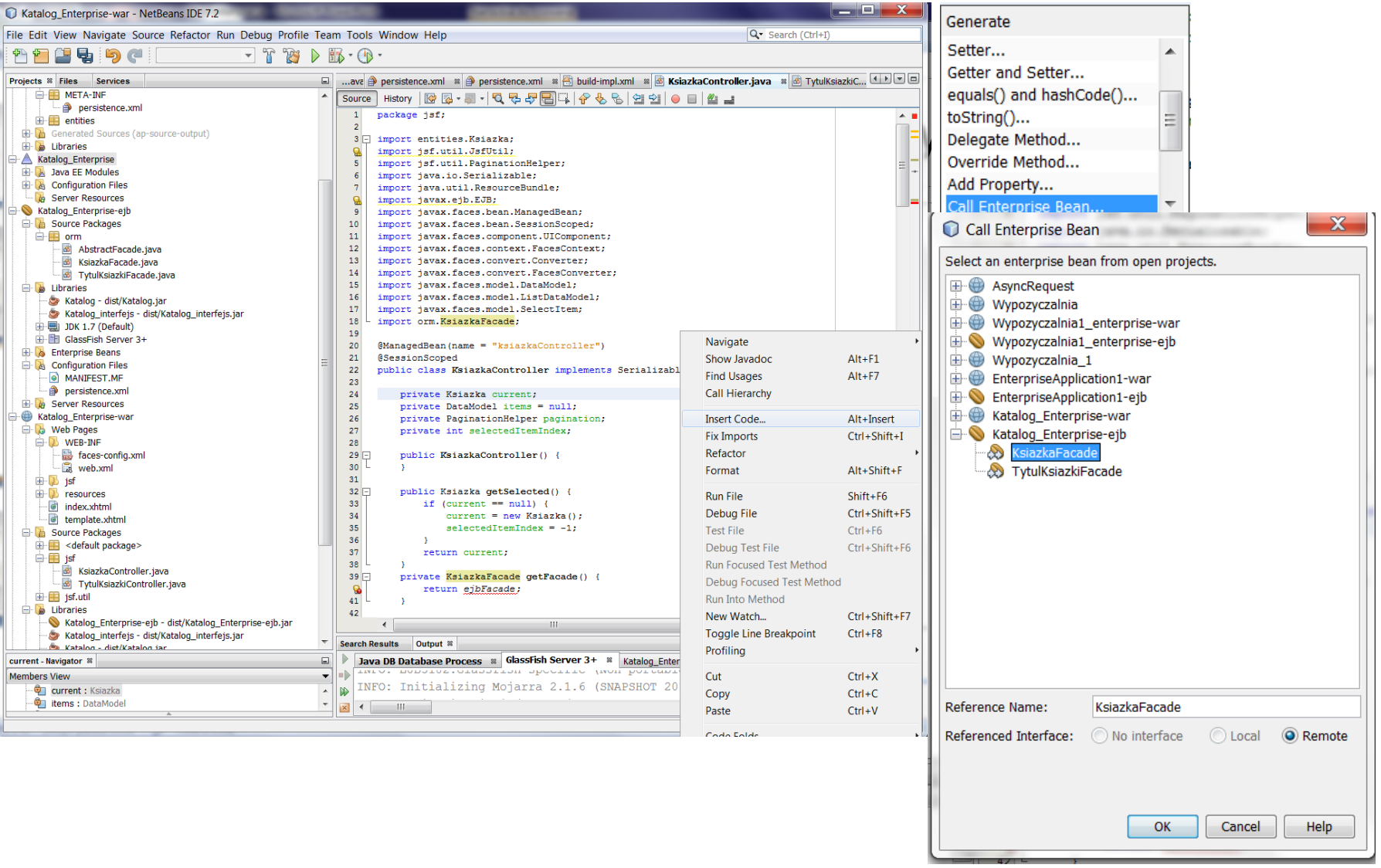
8.10. Poprawa importów w klasie TytulKsiazkiController



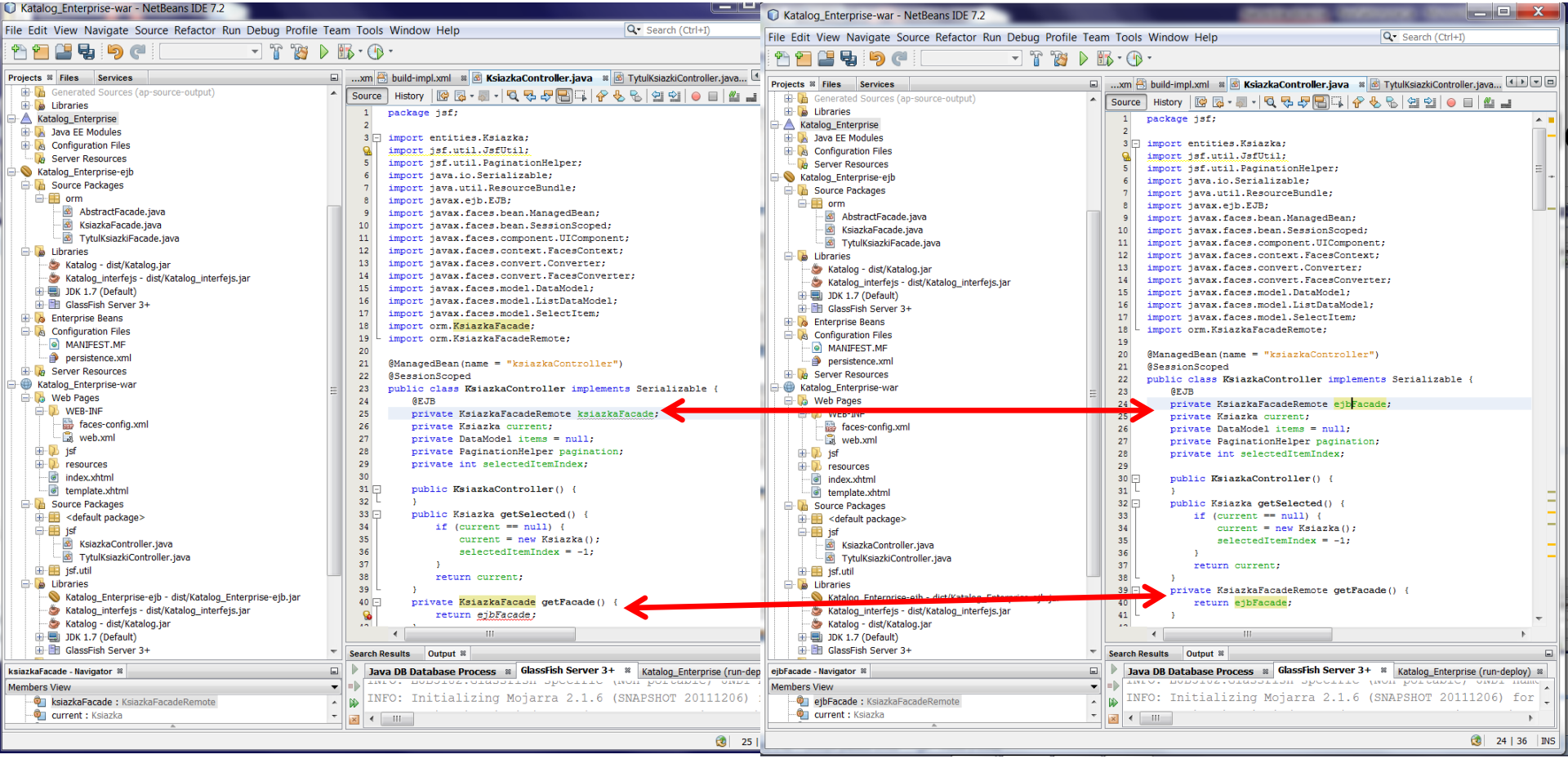
8.11. Usuwanie błędów kompilacji w klasie KsiazkaController - po kliknięciu prawym klawiszem na klasę KsiazkaController należy wybrać pozycję Fix Imports i wybrać właściwe klasy w okienku Fix All Imports.



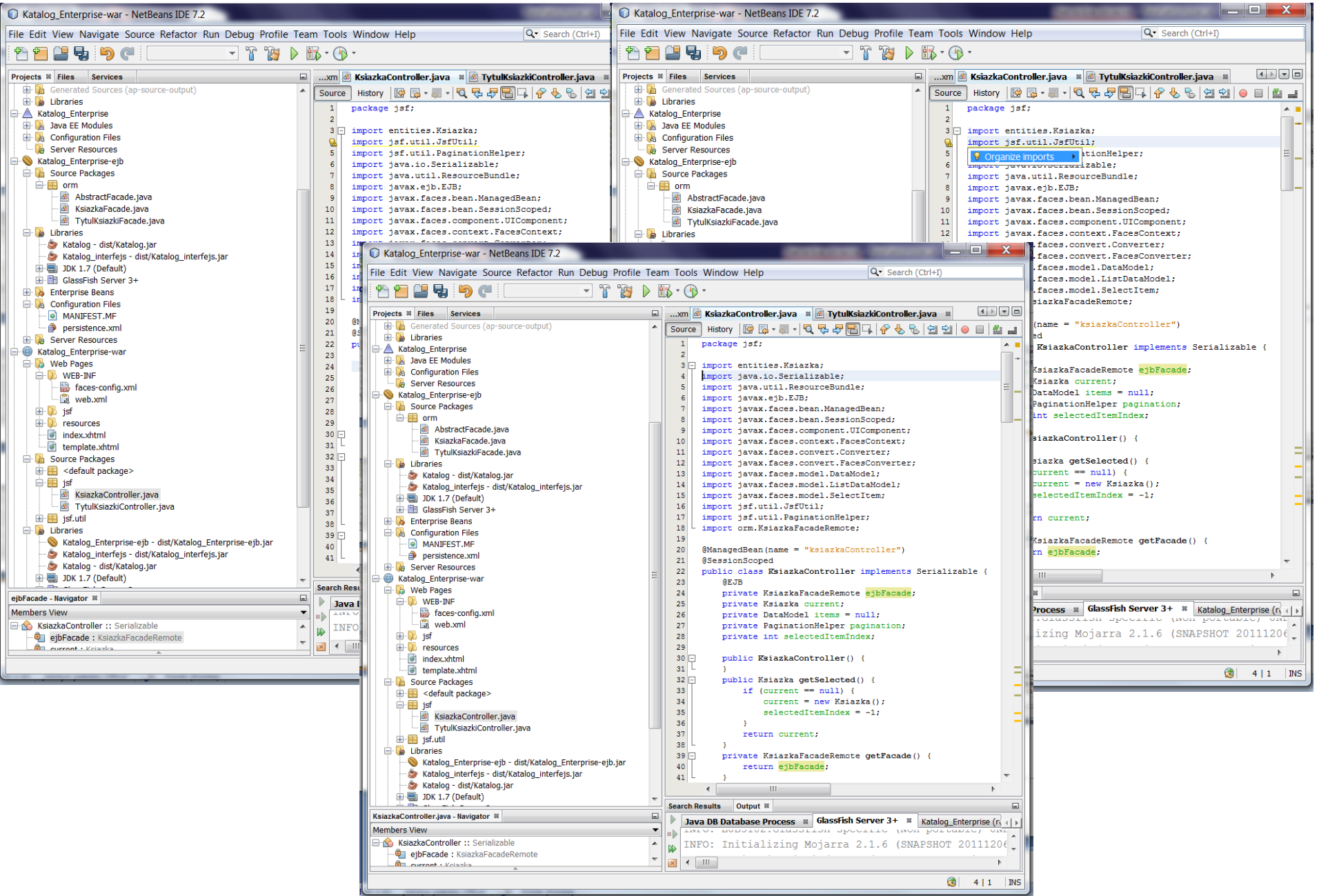
8.12. Dodanie dostępu do komponentów z warstwy integracji - po kliknięciu prawym klawiszem na klasę KsiazkaController należy wybrać pozycję Insert Code…, wybrać pozycję Call Enterprise Bean… w okienku Generate. Następnie, należy wybrać komponent typu KsiazkaFacade w oknie Select an enterprise bean from open projects.



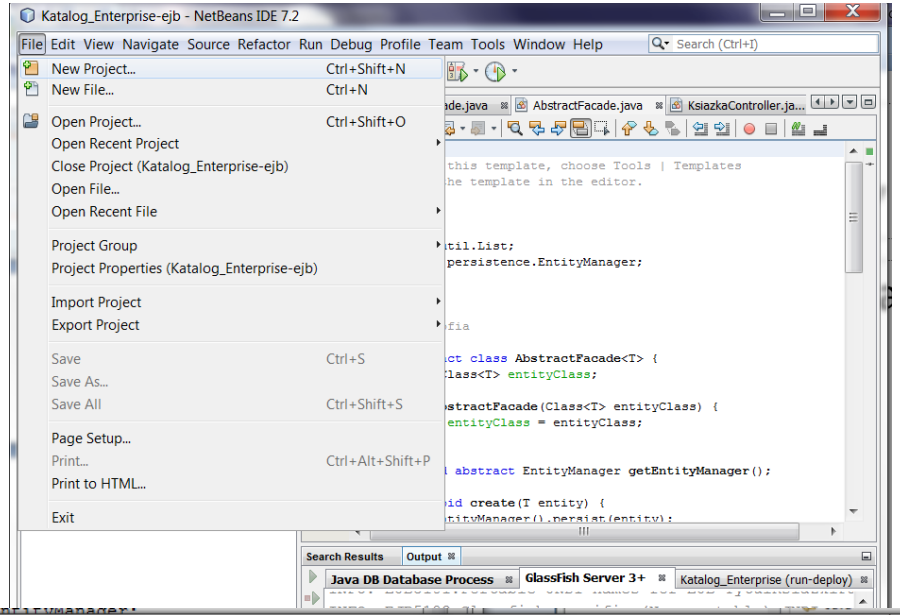
8.13. Poprawa kodu w klasie KsiazkaController wynikająca ze zmiany nazwy dodanego obiektu oraz zmiany typu obiektu dodanego za pomocą adnotacji.

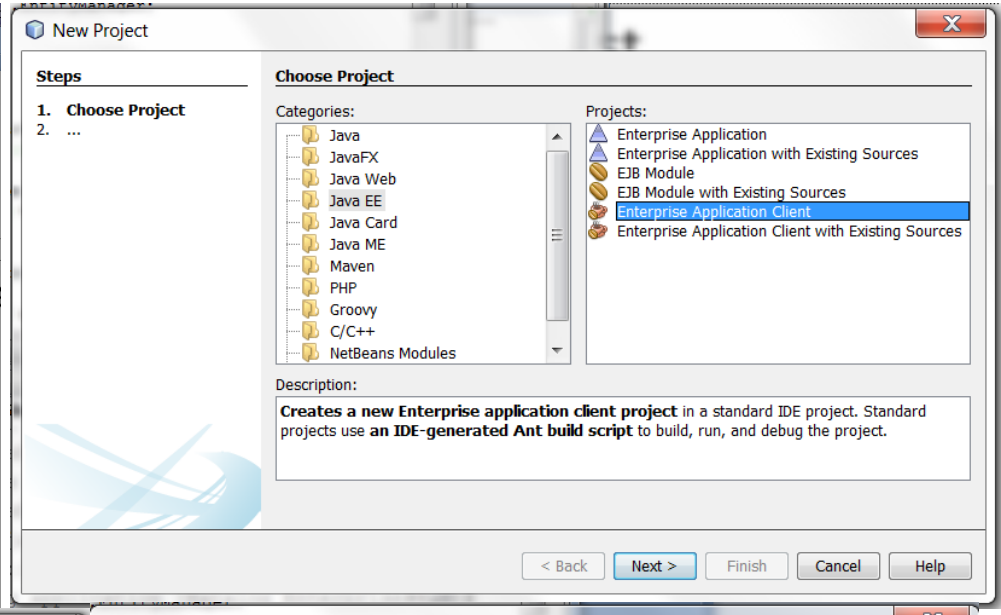


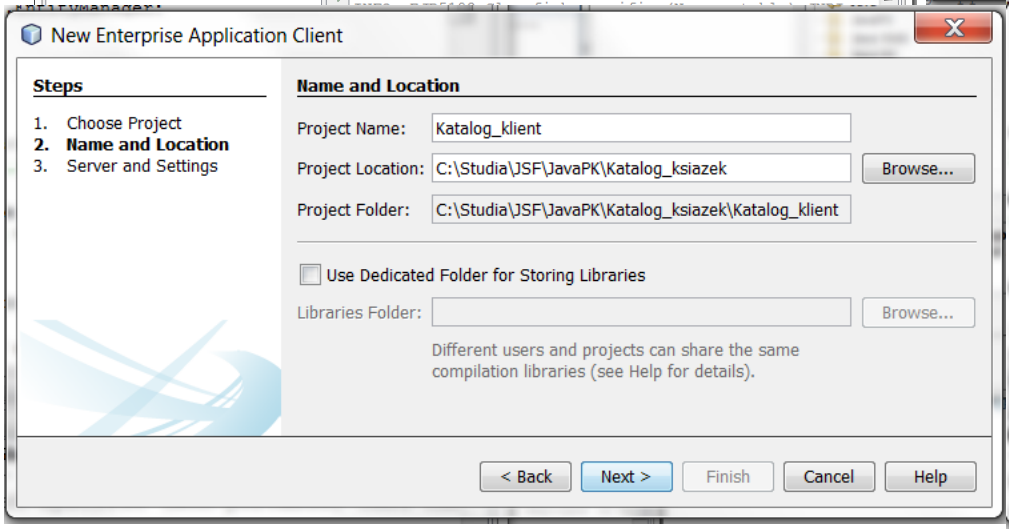
8.14. Poprawa importów w klasie KsiazkaController.

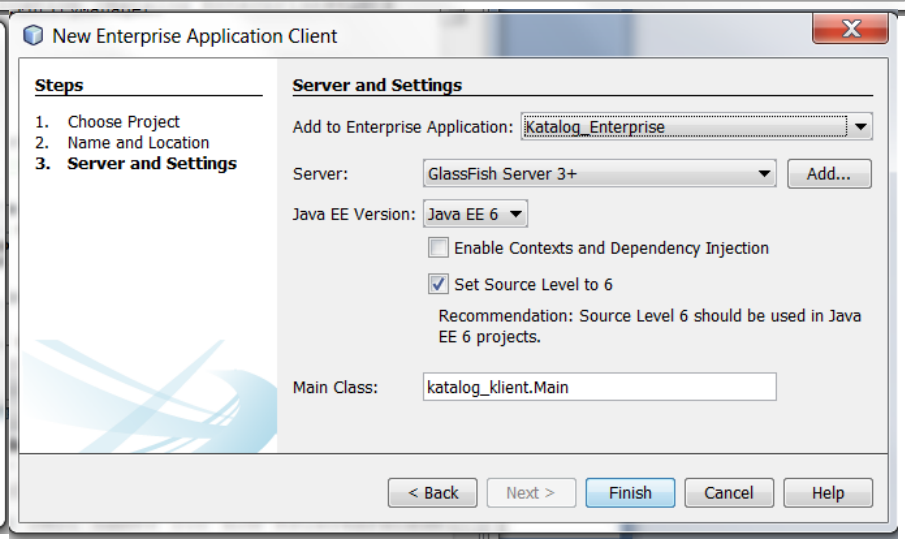


9.1. Utworzenie projektu klienta typu Enterprise : New Project/Java EE/Enterprise Application Client; Next; następnie należy nadać nazwę Katalog_klient (Project Name) w wybranym katalogu (Project Location); Next; następnie należy wybrać Project Katalog_Enterprise (Add to Enterprise Application), serwer GlassFish (Server) i platformę Java EE 6 (Java EE Version)

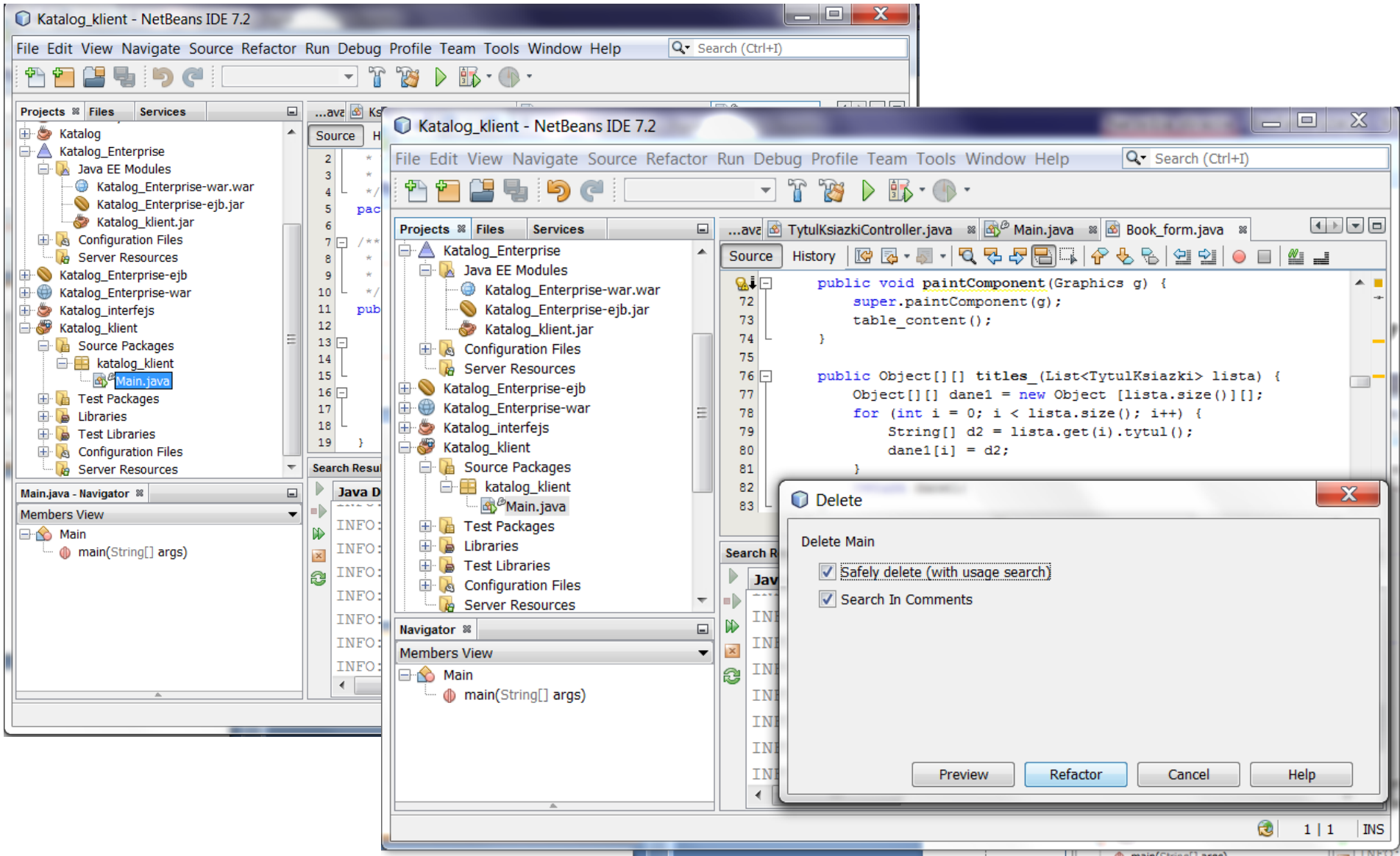








9.2. Należy usunąć klasę Main z projektu Katalog_klient, utworzoną domyślnie w domyślnym pakiecie katalog_klient



9.3. Należy do klasy TytulKsiazki w projekcie Katalog dodać metodę tytul() do prezentowania danych obiektu typu TytyulKsiazki

```
public String[] tytul()
   {
      String[] help={tytulId.toString(), tytul, autorNazwisko, autorImie,
                              isbn, wydawnictwo};
      return help;
   }
```

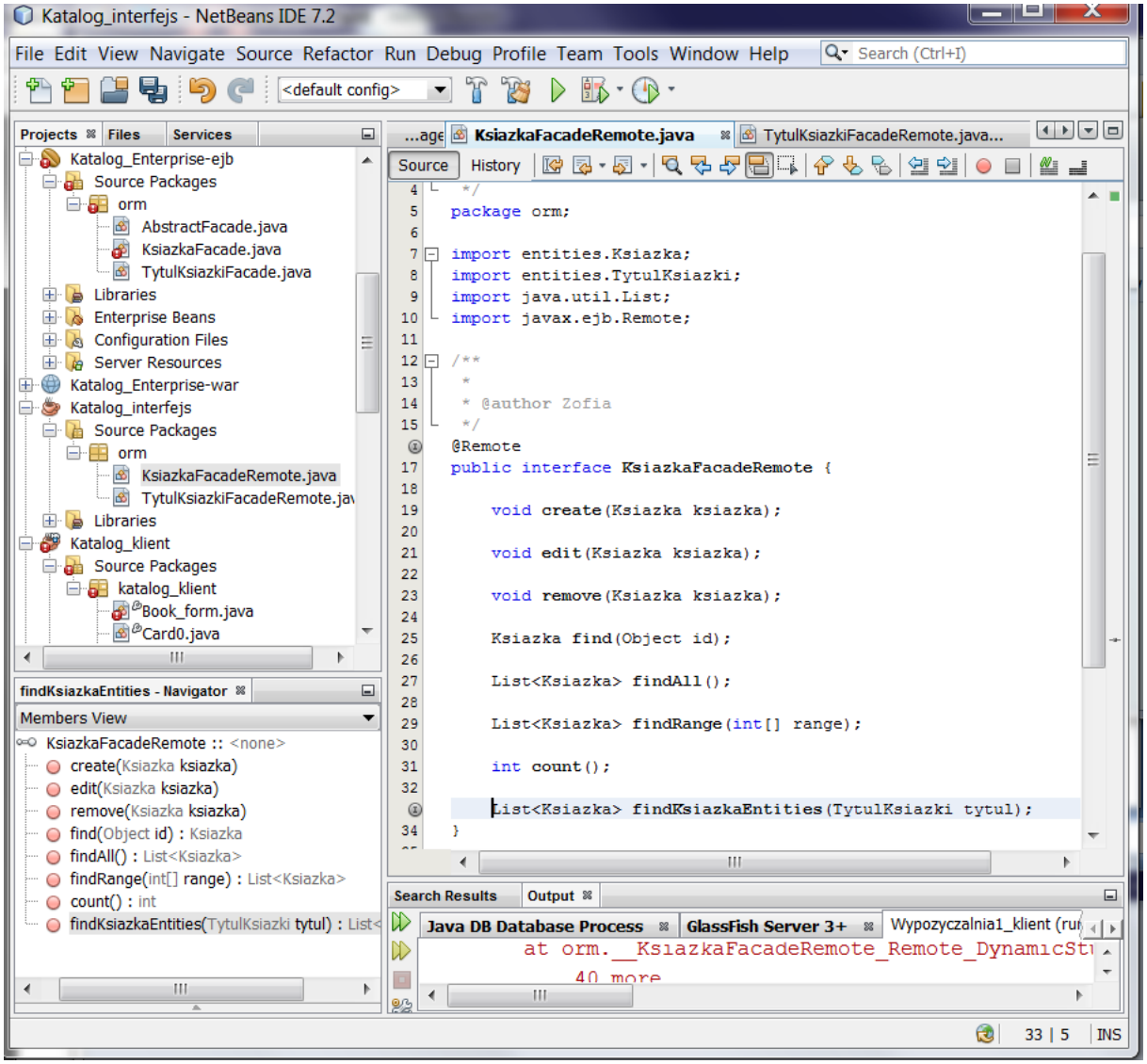
9.4. Należy do klasy Ksiazka w projekcie Katalog dodać metodę toString_() do prezentowania danych obiektu typu Ksiazka

```
public String toString_() {
    return "entities.Ksiazka[ ksiazkaId=" + ksiazkaId + " ] " +idTytul.toString()+
            " numer: "+numer;
  }
```

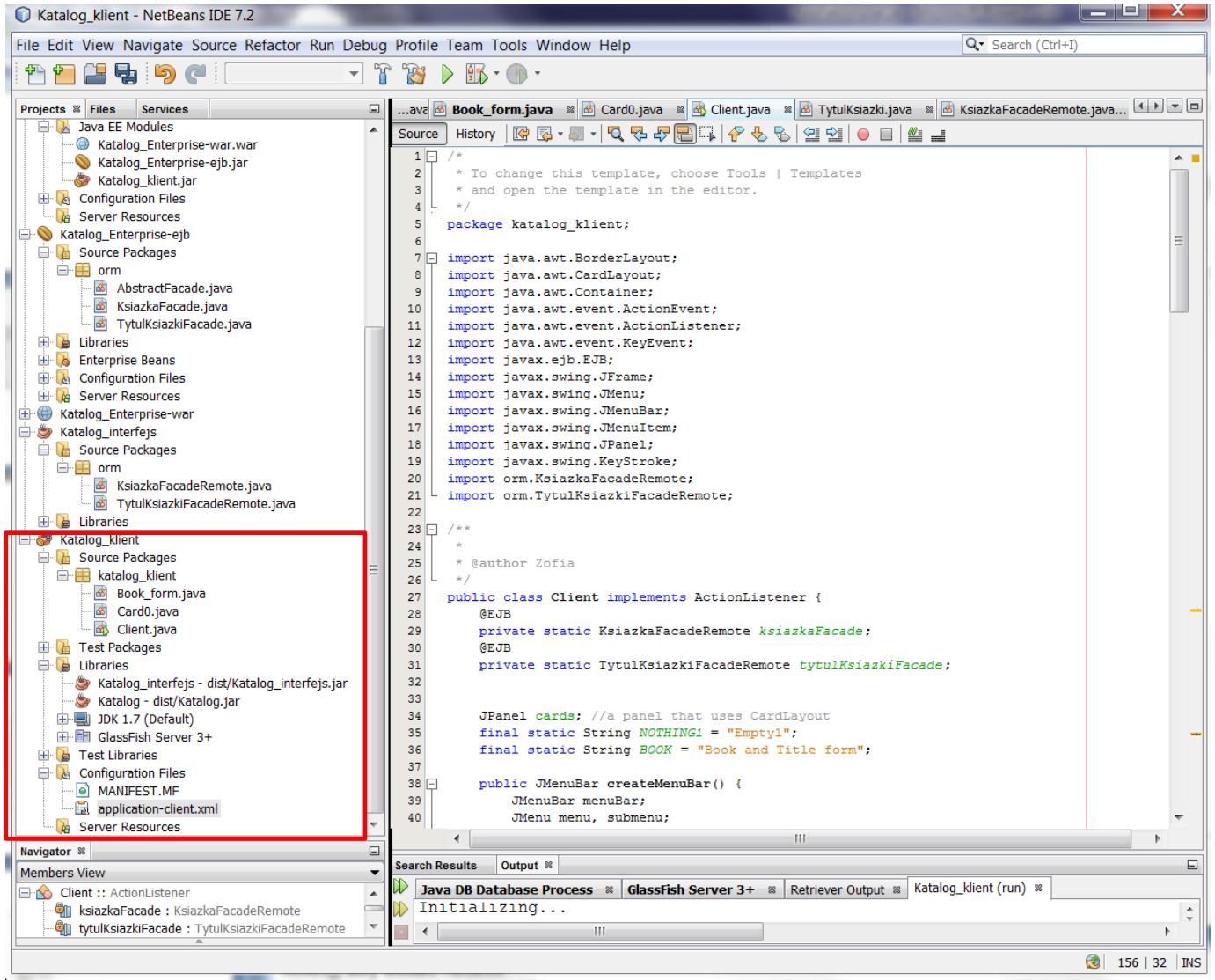
9.5. Należy do klasy KsiazkaFacade w projekcie Katalog dodać metodę findKsiazkaEntities do pobrania danych typu Ksiazka obiektu typu TytulKsiazki

```
@Override
  public List<Ksiazka> findKsiazkaEntities(TytulKsiazki tytul) {
     Query q = em.createQuery("SELECT k FROM Ksiazka k WHERE
                                          k.idTytul.tytulId = " + tytul.getTytulId());
     return q.getResultList();
  }
```

9.6. Uzupełnienie definicji interfejsu KsiazkaFacadeRemote o metodę findKsiazkaEntities



9.7. Uzupełnie głównej klasy Client o dwie adnotacje do obu obiektów typu Session Bean for Entity class: TytulKsiazkiFacade oraz KsiazkaFacade (prawym klawiszem należy kliknąć na kod klasy Client, następnie wybrać Insert Code.., i potem wybrać z listy formularza Call Enterprise Bean obiekty typu Session Bean for Entity class: TytulKsiazkiFacade oraz KsiazkaFacade)



9.8. Kod klas z pakietu katalog_klient z projektu Katalog_klient: Book_form

```
package katalog_klient;

import entities.Ksiazka;
import entities.TytulKsiazki;
import java.awt.Dimension;
import java.awt.Graphics;
import java.util.Iterator;
import java.util.List;
import javax.swing.BoxLayout;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.table.AbstractTableModel;

public class Book_form extends JPanel {

    private JTable table;
    int row = 0;
    Client client;
    MyTableModel model;
    JComboBox books;
```

```
public Book_form(Client client) {
    super();
    this.client = client;
    setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
    model = new MyTableModel();
    table_content();
    table = new JTable(model);
    table.setPreferredScrollableViewportSize(new Dimension(500, 100));
    table.setFillsViewportHeight(true);
    table.getSelectionModel().addListSelectionListener(new RowListener());

    add(new JScrollPane(table));

    JLabel lbooks = new JLabel("Books");
    add(lbooks);
    books = new JComboBox();
    add(books);
  }

  @Override
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    table_content();
  }
```

```
//wyswietlanie tytulow
   void table_content() {
      List<TytulKsiazki> titles = Client.getTytulKsiazkiFacade().findAll();
      model.setData(titles_(titles));
   }

   public Object[][] titles_(List<TytulKsiazki> lista) {
      Object[][] dane1 = new Object[lista.size()][];
      for (int i = 0; i < lista.size(); i++) {
         String[] d2 = lista.get(i).tytul();
         dane1[i] = d2;
      }
      return dane1;
   }

   // wyswietlanie książek podanego tytulu
   void print_books() {
      TytulKsiazki tytul = Client.getTytulKsiazkiFacade().find(title_id());
      List<Ksiazka> ksiazki = Client.getKsiazkaFacade().findKsiazkaEntities(tytul);
      if (ksiazki == null) {
         return;
      }
      list_content(ksiazki, books);
   }

   int title_id() {
      String data = (String) model.getValueAt(row, 0);
      return Integer.parseInt(data);
   }

   private void list_content(List<Ksiazka> ksiazki, JComboBox list) {
      String s;
      list.removeAllItems();
      Iterator<Ksiazka> iterator = ksiazki.iterator();
      while (iterator.hasNext()) {
         s = iterator.next().toString_();
         list.addItem(s);
      }
   }
```

```
private class RowListener implements ListSelectionListener {

    @Override
    public void valueChanged(ListSelectionEvent event) {
       if (event.getValueIsAdjusting()) {
          return;
       }
       row = table.getSelectionModel().getLeadSelectionIndex();
       print_books();
    }
 }

  class MyTableModel extends AbstractTableModel {

    private String[] columnNames = {"ID", "Title",
       "First Name",
       "Name",
       "ISBN",
       "Publisher"};
    private Object[][] data;

    public void setData(Object[][] val) {
       data = val;
    }

    @Override
    public int getColumnCount() {
       return columnNames.length;
    }

    @Override
    public int getRowCount() {
       return data.length;
    }

    @Override
    public String getColumnName(int col) {
       return columnNames[col];
    }
```

```
 @Override
      public Object getValueAt(int row, int col) {
         return data[row][col];

      }

      @Override
      public Class getColumnClass(int c) {
         return getValueAt(0, c).getClass();
      }

      @Override
      public boolean isCellEditable(int row, int col) {
         if (col < 0) {
            return false;
         } else {
            return true;
         }
      }

      @Override
      public void setValueAt(Object value, int row, int col) {
         data[row][col] = value;
         fireTableCellUpdated(row, col);
      }
   }
}
```

9.9. Kod klas z pakietu katalog_klient z projektu Katalog_klient: Client

```
package katalog_klient;

import java.awt.BorderLayout;
import java.awt.CardLayout;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import javax.ejb.EJB;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.KeyStroke;
import orm.KsiazkaFacadeRemote;
import orm.TytulKsiazkiFacadeRemote;

public class Client implements ActionListener {
    @EJB
    private static KsiazkaFacadeRemote ksiazkaFacade;
    @EJB
    private static TytulKsiazkiFacadeRemote tytulKsiazkiFacade;


    JPanel cards; //a panel that uses CardLayout
    final static String NOTHING1 = "Empty1";
    final static String BOOK = "Book and Title form";
```

```
public JMenuBar createMenuBar() {
    JMenuBar menuBar;
    JMenu menu, submenu;
    JMenuItem menuItem;

    //Create the menu bar.
    menuBar = new JMenuBar();

    menu = new JMenu("A Menu");
    menu.setMnemonic(KeyEvent.VK_A);
    menuBar.add(menu);

    menuItem = new JMenuItem(BOOK);
    menuItem.setMnemonic(KeyEvent.VK_B);
    menuItem.addActionListener(this);
    menu.add(menuItem);

    menuItem = new JMenuItem(NOTHING1);
    menuItem.setMnemonic(KeyEvent.VK_E);
    menuItem.addActionListener(this);
    menu.add(menuItem);

    menu.addSeparator();

    submenu = new JMenu("A submenu");
    submenu.setMnemonic(KeyEvent.VK_S);

    menuItem = new JMenuItem(NOTHING1);
    menuItem.setAccelerator(KeyStroke.getKeyStroke(
          KeyEvent.VK_2, ActionEvent.ALT_MASK));
    menuItem.addActionListener(this);
    submenu.add(menuItem);

    menuItem = new JMenuItem(NOTHING1);
    menuItem.addActionListener(this);
    submenu.add(menuItem);
    menu.add(submenu);

    //Build second menu in the menu bar.
    menu = new JMenu("Another Menu");
    menu.setMnemonic(KeyEvent.VK_N);
    menuBar.add(menu);

    return menuBar;
  }
```

```
public static KsiazkaFacadeRemote getKsiazkaFacade() {
    return ksiazkaFacade;
  }

  public static void setKsiazkaFacade(KsiazkaFacadeRemote ksiazkaFacade) {
    Client.ksiazkaFacade = ksiazkaFacade;
  }

  public static TytulKsiazkiFacadeRemote getTytulKsiazkiFacade() {
    return tytulKsiazkiFacade;
  }

  public static void setTytulKsiazkiFacade(TytulKsiazkiFacadeRemote tytulKsiazkiFacade) {
    Client.tytulKsiazkiFacade = tytulKsiazkiFacade;
  }

  public Container createContentPane() {
    //Create the content-pane-to-be.

    Card0 card0 = new Card0();

    Book_form card2 = new Book_form(this);


    //Create the panel that contains the "cards".
    cards = new JPanel(new CardLayout());
    cards.add(card0, NOTHING1);

    cards.add(card2, BOOK);


    JPanel p1 = new JPanel();

    p1.add(cards, BorderLayout.CENTER);
    return p1;
  }
```

```
 public void actionPerformed(ActionEvent e) {

       JMenuItem source = (JMenuItem) (e.getSource());
       CardLayout cl = (CardLayout) (cards.getLayout());
       if (source.getText().equals(BOOK)) {
          cl.show(cards, BOOK);
       } else if (source.getText().equals(NOTHING1)) {
          cl.show(cards, NOTHING1);
       }
    }

    /**
     * Create the GUI and show it. For thread safety, this method should be
     * invoked from the event-dispatching thread.
     */
    private static void createAndShowGUI() {
       //Create and set up the window.
       JFrame frame = new JFrame("MenuDemo");
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       frame.setSize(800, 460);
       //Create and set up the content pane.
       Client demo = new Client();
       frame.setJMenuBar(demo.createMenuBar());
       frame.setContentPane(demo.createContentPane());

       //Display the window.
       frame.setVisible(true);
    }

    public static void main(String[] args) {
       //Schedule a job for the event-dispatching thread:
       //creating and showing this application's GUI.
       java.awt.EventQueue.invokeLater(new Runnable() {
          //javax.swing.SwingUtilities.invokeLater(new Runnable() {
          public void run() {
             createAndShowGUI();
          }
       });
    }
}
```

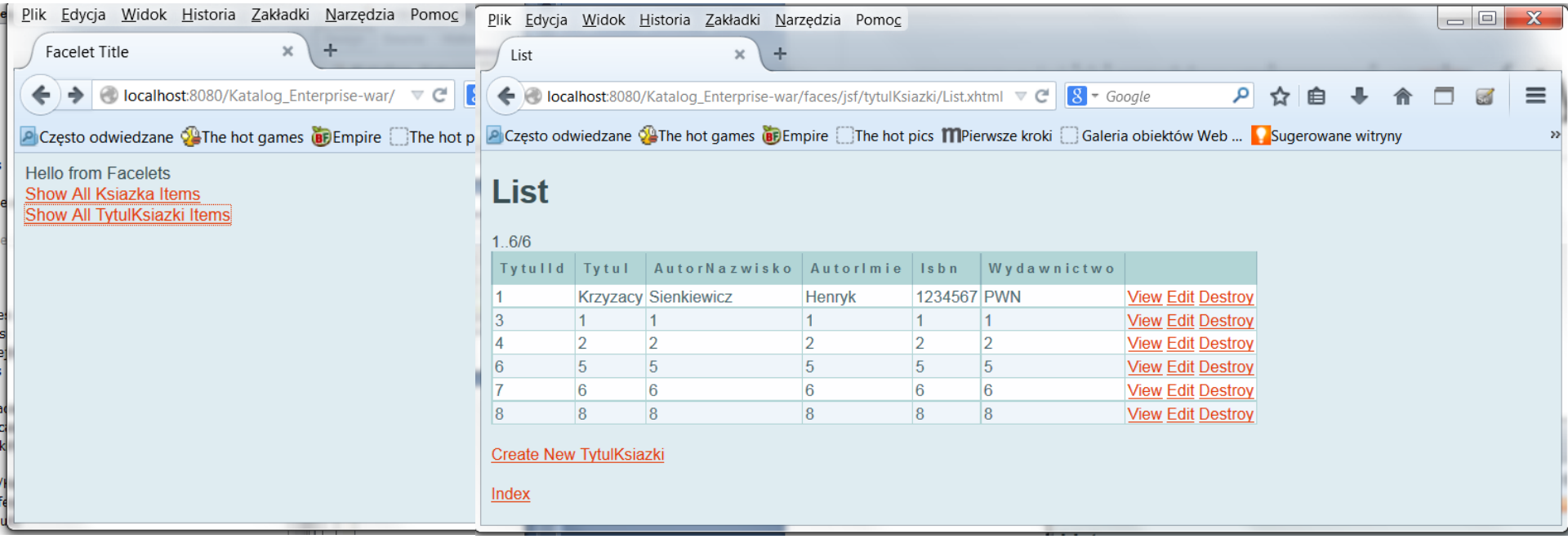
9.9. Kod klas z pakietu katalog_klient z projektu Katalog_klient: Card0

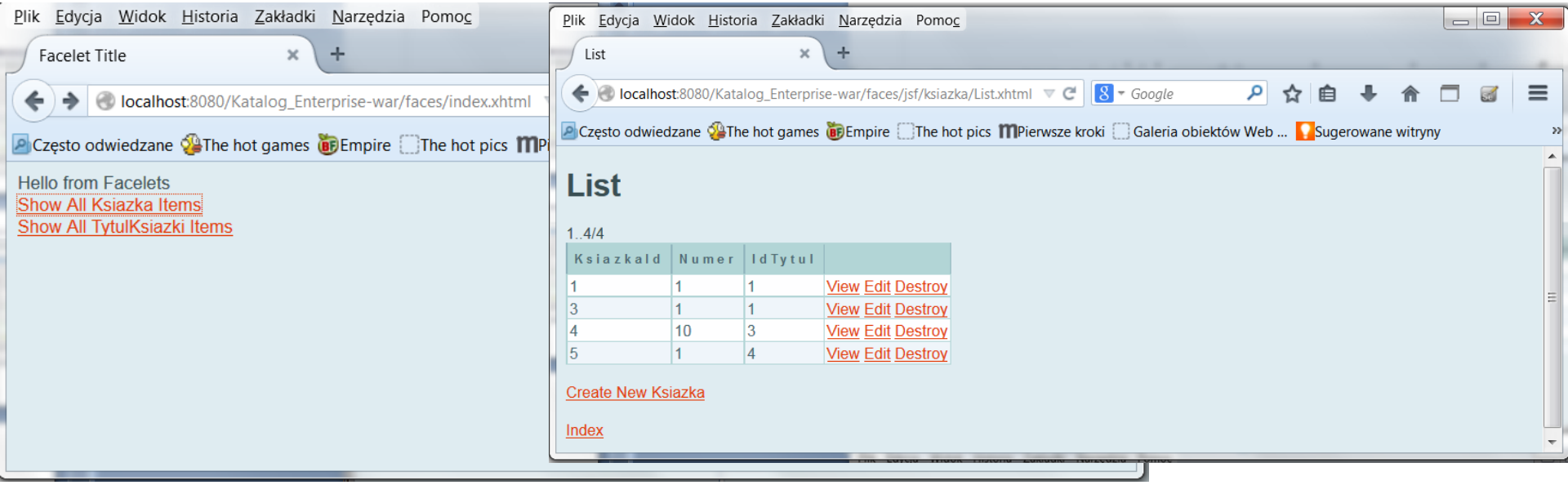
```
package katalog_klient;

import javax.swing.JPanel;

public class Card0 extends JPanel{
    public Card0() {
    }

}
```

10. Uruchomienie aplikacji

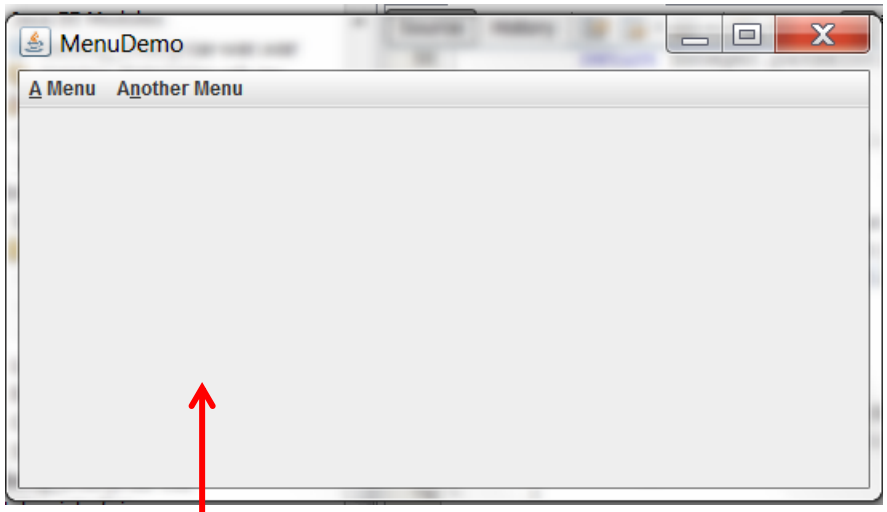
1. Wykonanie operacji Clean/Build w zakładce Projects dla projektu Katalog
2. Wykonanie operacji Clean/Build w zakładce Projects dla projektu Katalog_interfejs.
3. Wykonanie operacji Clean/Build w zakładce Projects dla projektu Katalog_Enterprise-ejb.
4. Wykonanie operacji Clean/Build w zakładce Projects dla projektu Katalog_Enterprise-war.
5. Wykonanie operacji Clean/Build w zakładce Projects dla projektu Katalog_klient.
6. Wykonanie operacji run dla projektu Katalog_Enterprise – w oknie domyślnej przeglądarki powinna uruchomić się aplikacja klienta internetowego (slajd 56).
7. Wykonanie operacji run dla klienta typu Enterprise Katalog_klient (slajd 57).

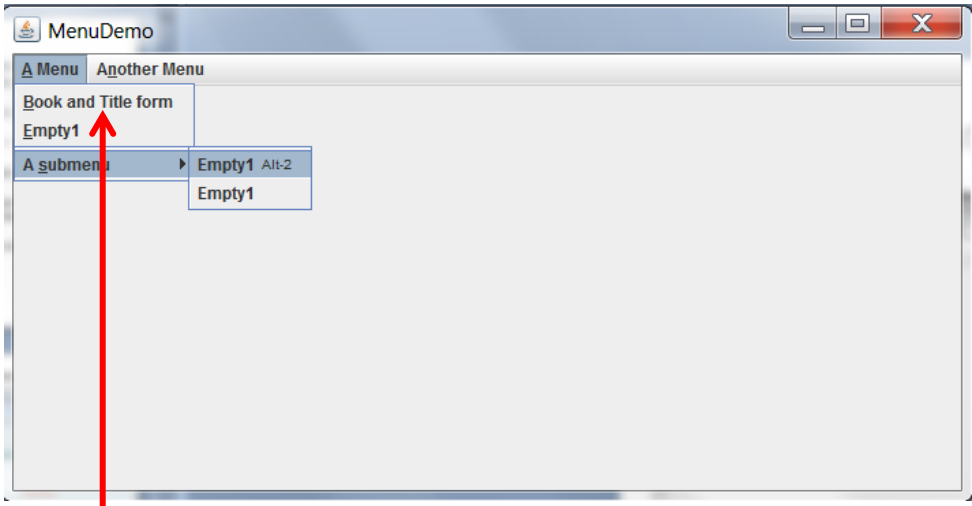
10.1. Widok formularzy klienta internetowego



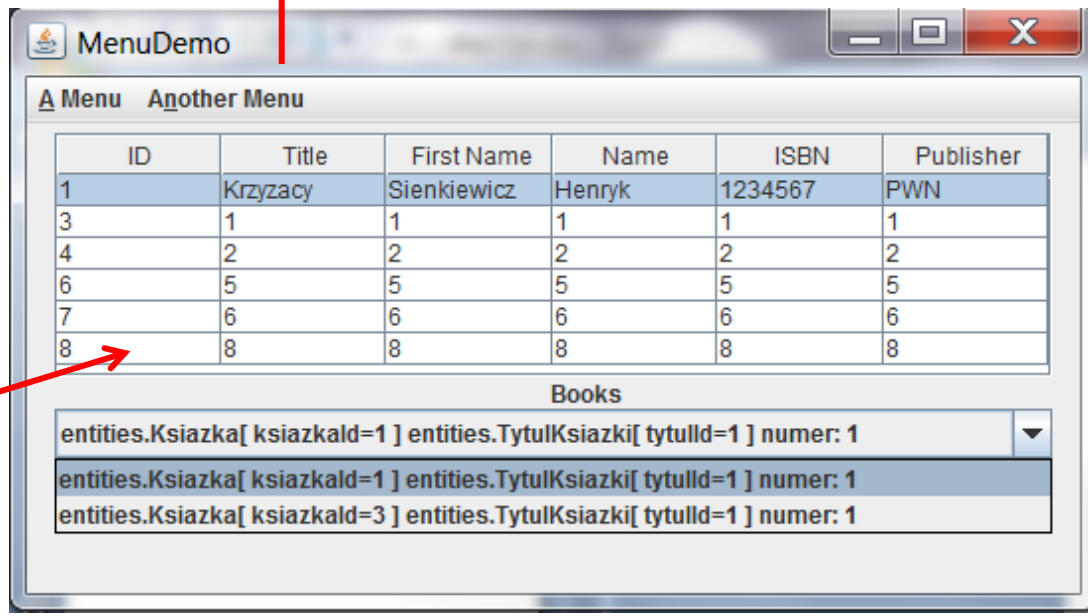


10.1. Widok formularzy klienta Enterprise





Widok formularza Card0



Widok formularza Book_form

11. Dalsze prace - propozycja

1. Wprowadzenie nowego szablonu strony internetowej
2. Wprowadzenie obiektów transferowych do przesyłania danych między warstwą klienta Enterprise i warstwą integracji oraz warstwą prezentacji klienta internetowego i warstwą integracji.