Budowa warstwy klienta w architekturze typu klient-serwer zbudowanych z komponentów typu EE - klient desktopowy i internetowy. Zastosowanie komponentów opartych na technologii EJB 3.2.
na podstawie https://docs.oracle.com/javaee/7/JEETT.pdf

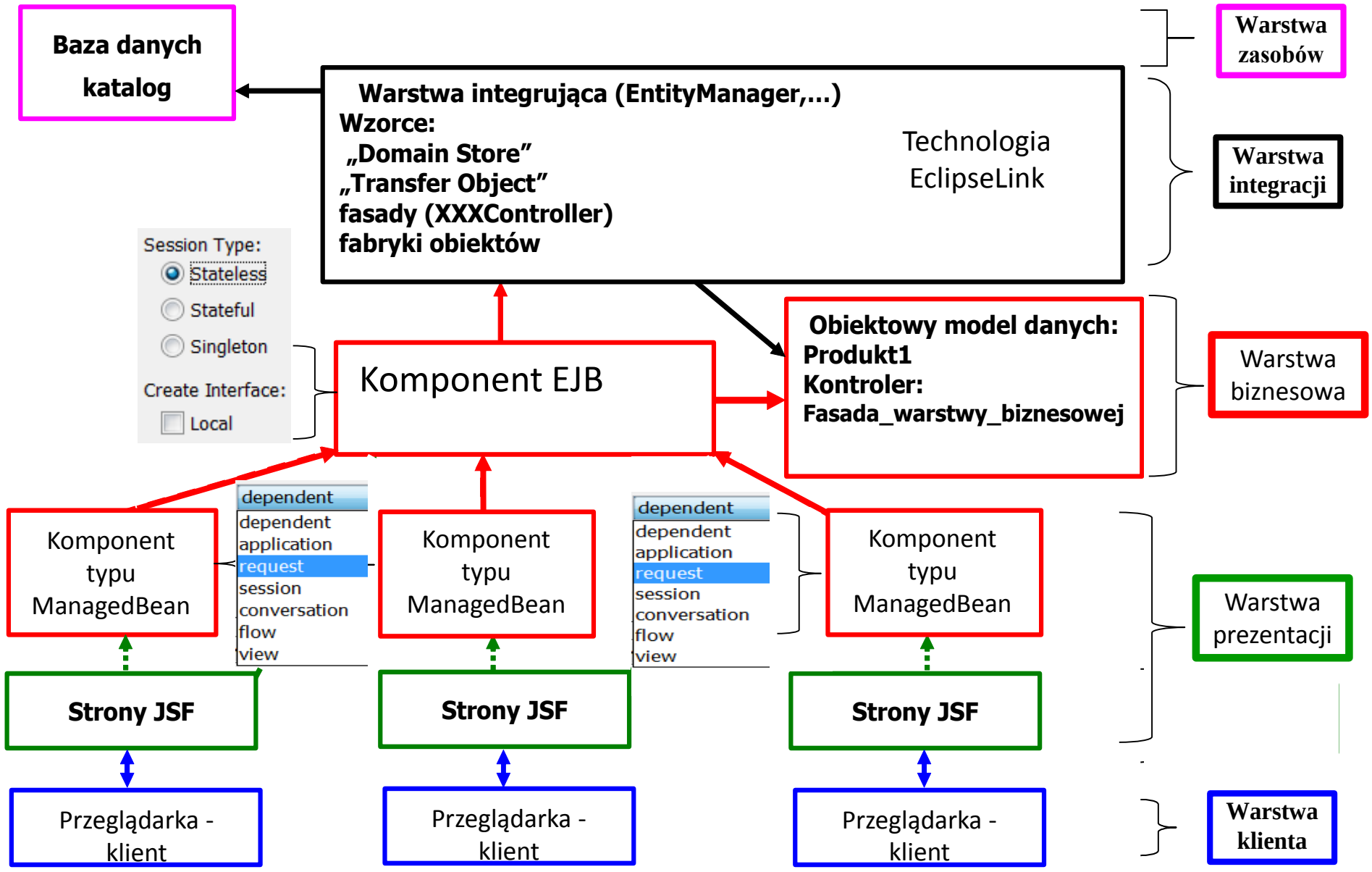
Programowanie komponentowe 5

1. Aplikacja wielowarstwowa Java EE oparta na komponencie EJB typu Session Bean – rodzaj Stateless lub Singleton

1.1. Warstwa internetowa oparta na komponencie Managed Bean o czasie życia **RequestScope**

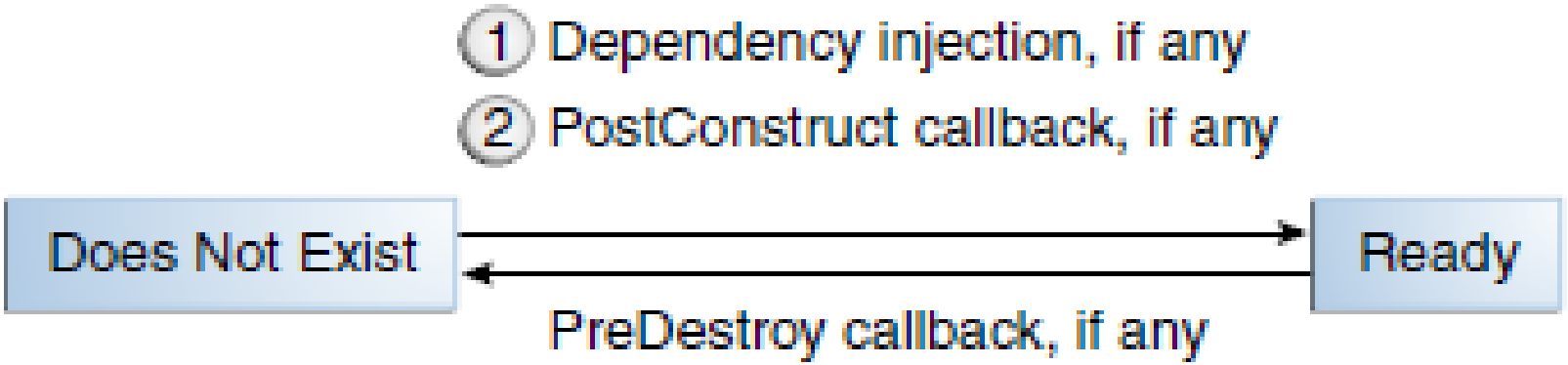
1.2. Warstwa internetowa oparta na komponencie Managed Bean o czasie życia **SessionScope**

Ad. 1.1. Architektura aplikacji pięciowarstwowej -Java EE 7.0 JavaServer Faces



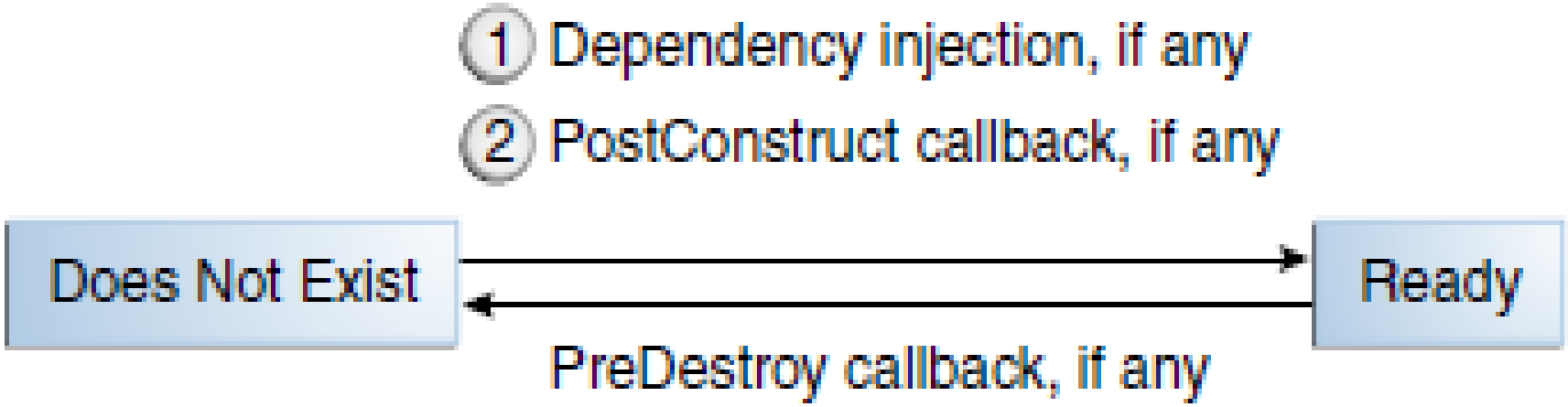
Cykl życia bezstanowych (Stateless) komponentów typu Session Bean – wykład 2

Ponieważ bezstanowy komponent typu Session Bean nigdy nie przechodzi w stan pasywny, cykl życia składa sie tylko z dwóch stanów: **nonexistent and ready** podczas wywoływania metod przez warstwę klienta.



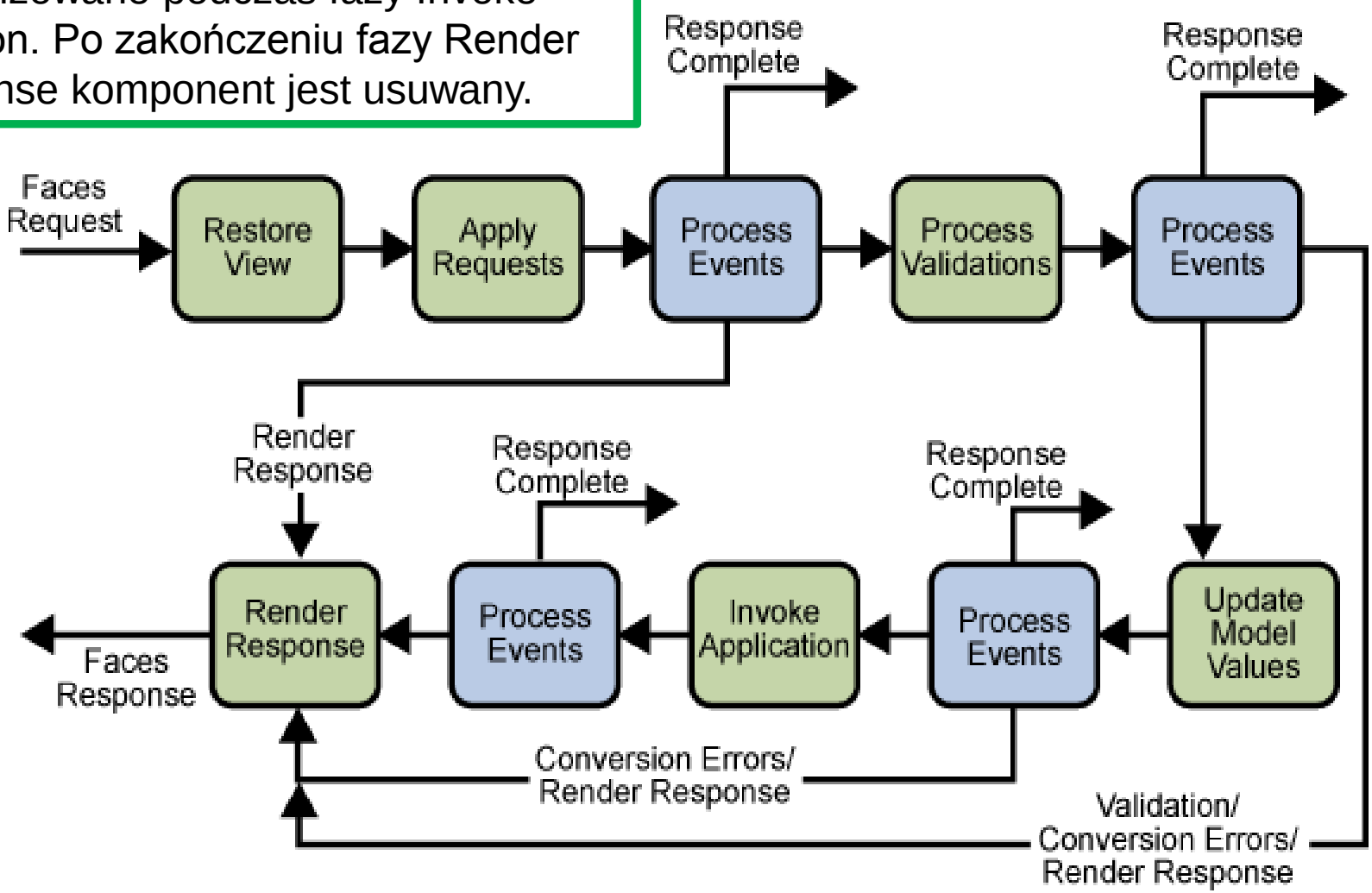
Cykl życia komponentów Singleton typu Session Bean

Ponieważ bezstanowy komponent typu Session Bean nigdy nie przechodzi w stan pasywny, cykl życia składa sie tylko z dwóch stanów: **nonexistent and ready** podczas wywoływania metod przez warstwę klienta. Singleton startuje razem z aplikacją, jeśli ma adnotację @**StartUp** i inicjuje aplikację za pomocą metody @PostConstruct (jeśli jest) i „czyści” za pomocą metody @PreDestroy (jeśli jest).



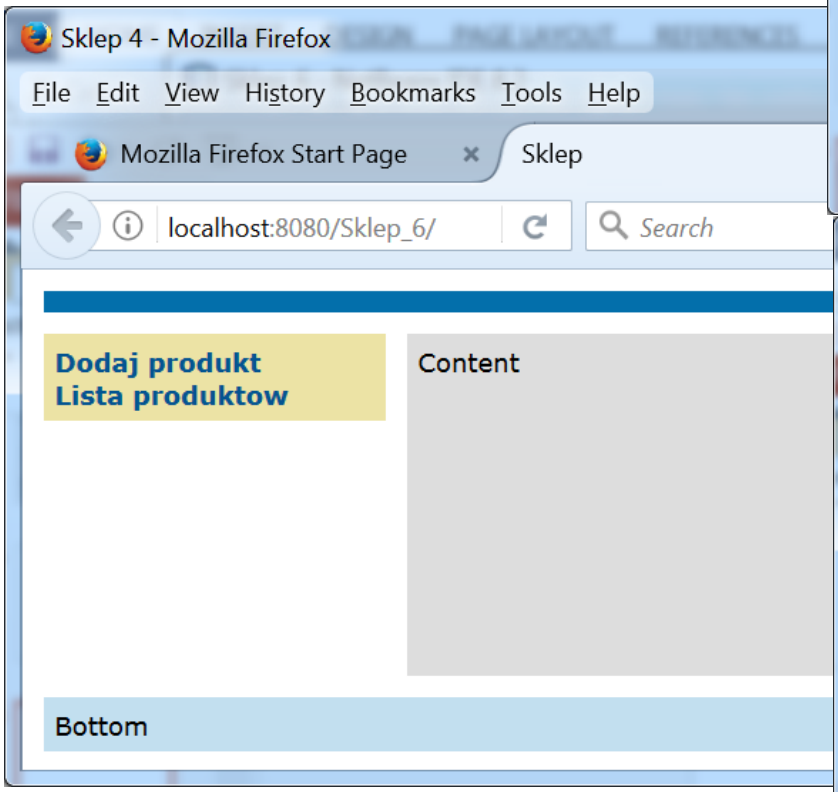
Standard cyklu życia „Request-Response” dla JavaServer Faces

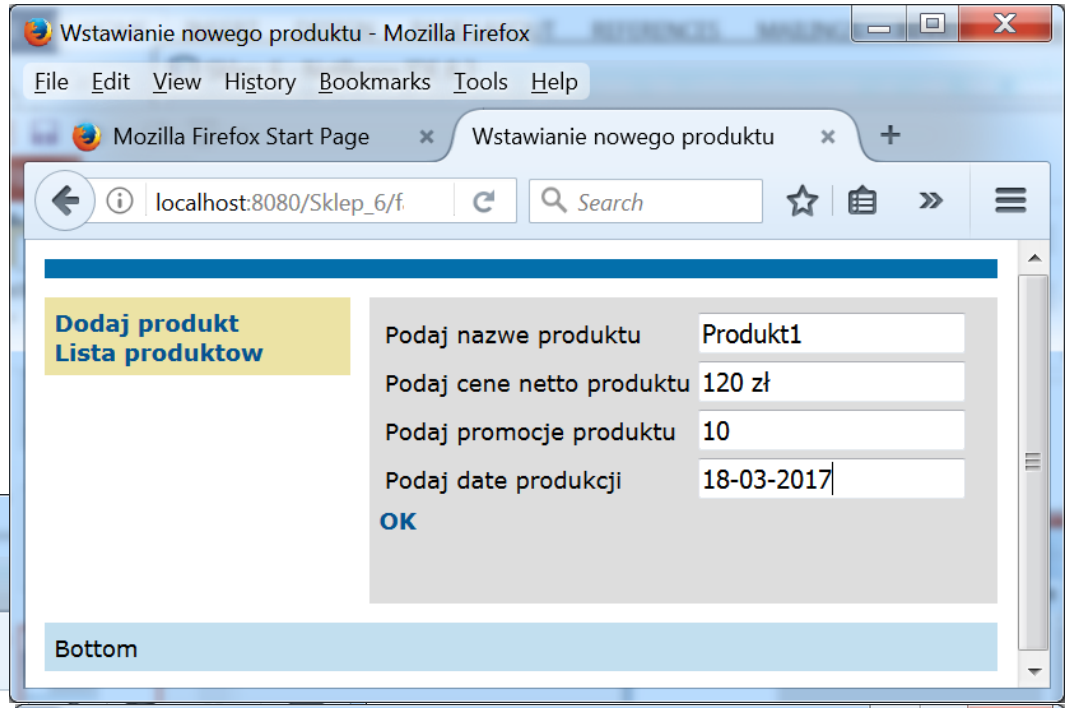
RequestScoped: podczas odebrania żądania tworzony jest nowy komponent typu **Managed_produkt** – wszystkie jego dane są zaktualizowane podczas fazy Invoke Application. Po zakończeniu fazy Render Response komponent jest usuwany.

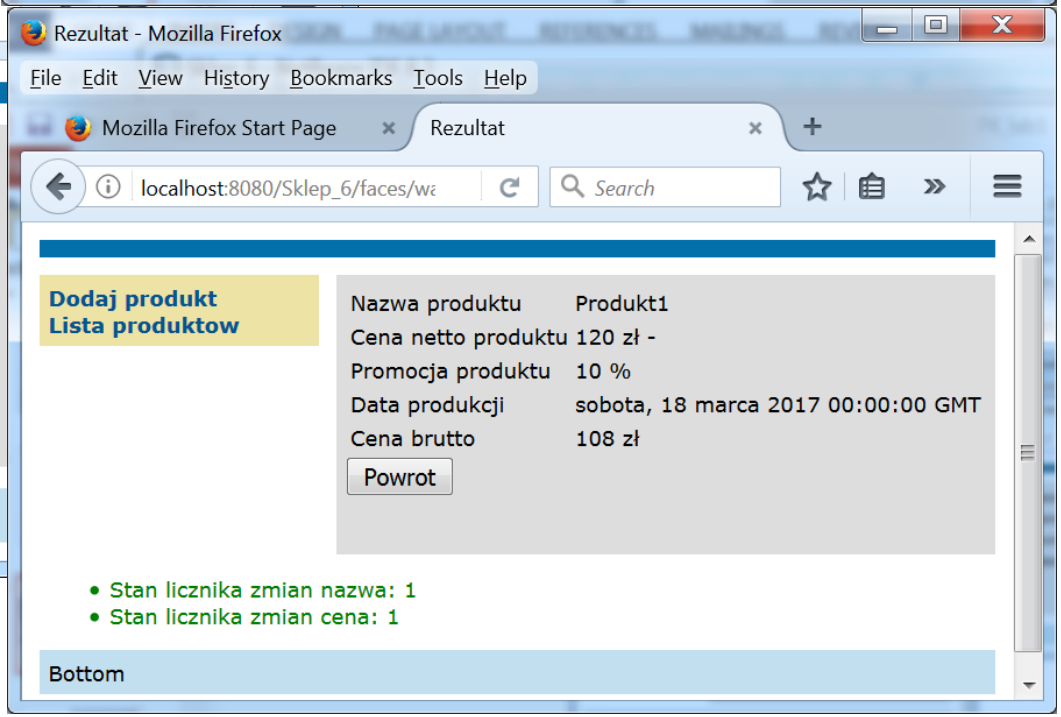


Uruchomienie aplikacji opartej na **komponencie EJB typu Session, rodzaj Stateless** - Clean and Build/Deploy/Run

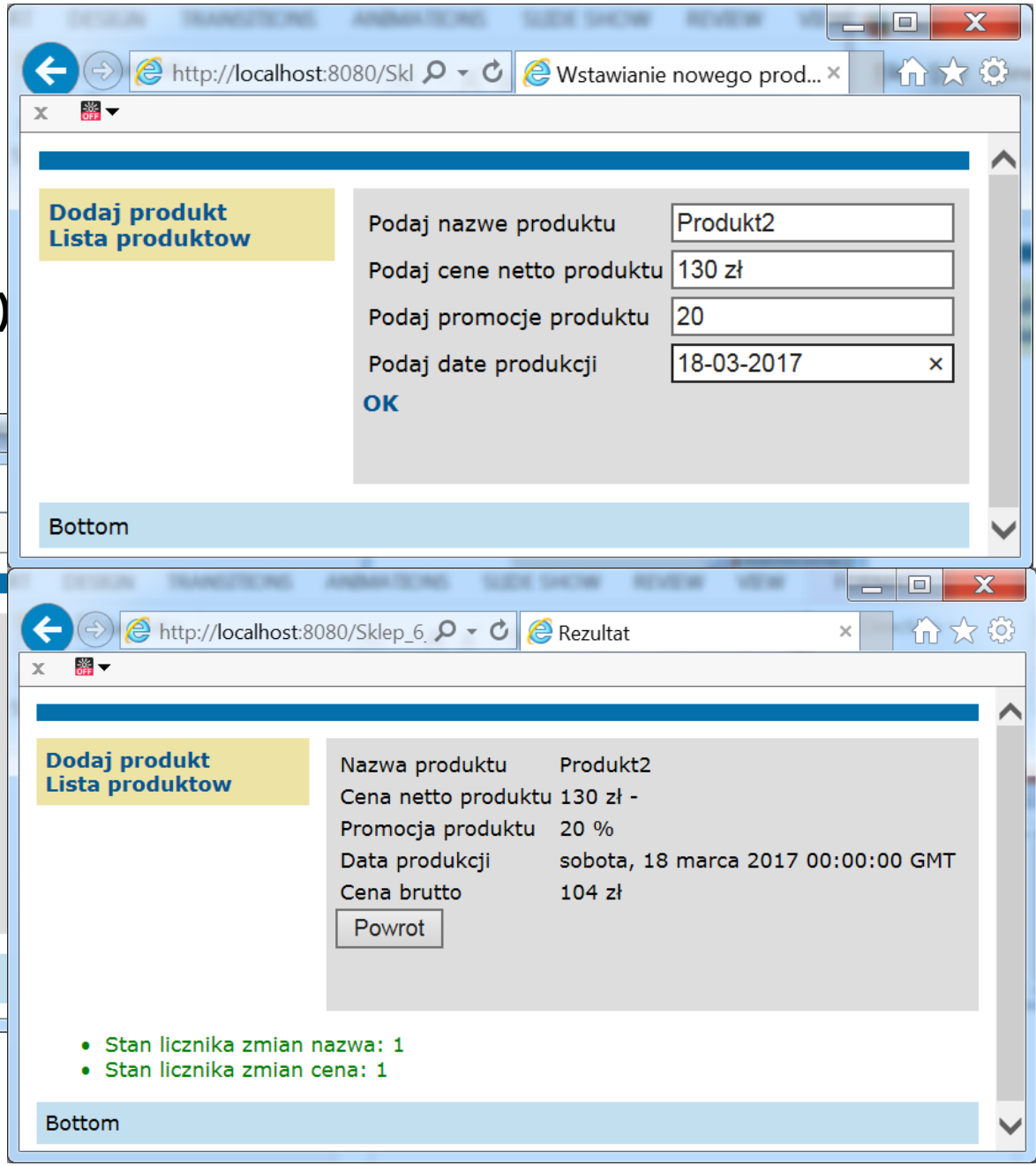
Uruchomienie 1-ej instancji warstwy klienta aplikacji i wprowadzenie danych produktu – może być uruchomiona automatycznie w przeglądarce domyślnej.

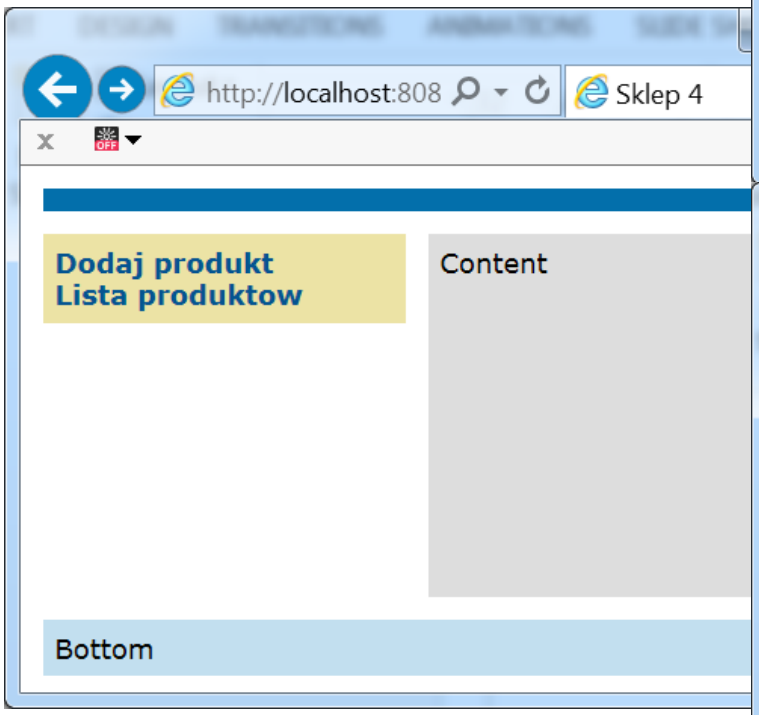




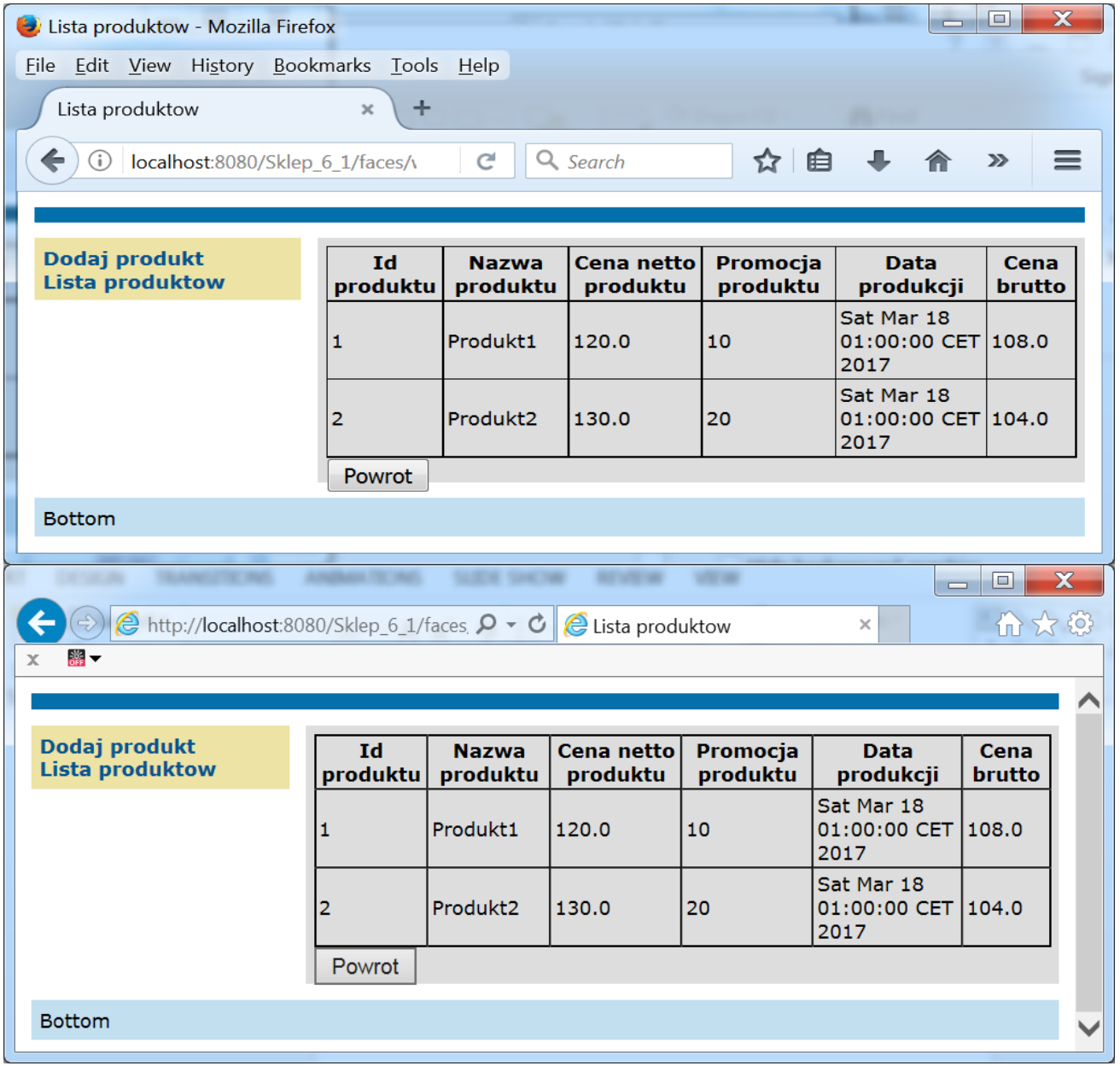


Uruchomienie 2-ej instancji warstwy klienta aplikacji aplikacji i wprowadzenie danych kolejnego produktu (przez wprowdzenie adresu: http://localhost:8080/Sklep_6_1)



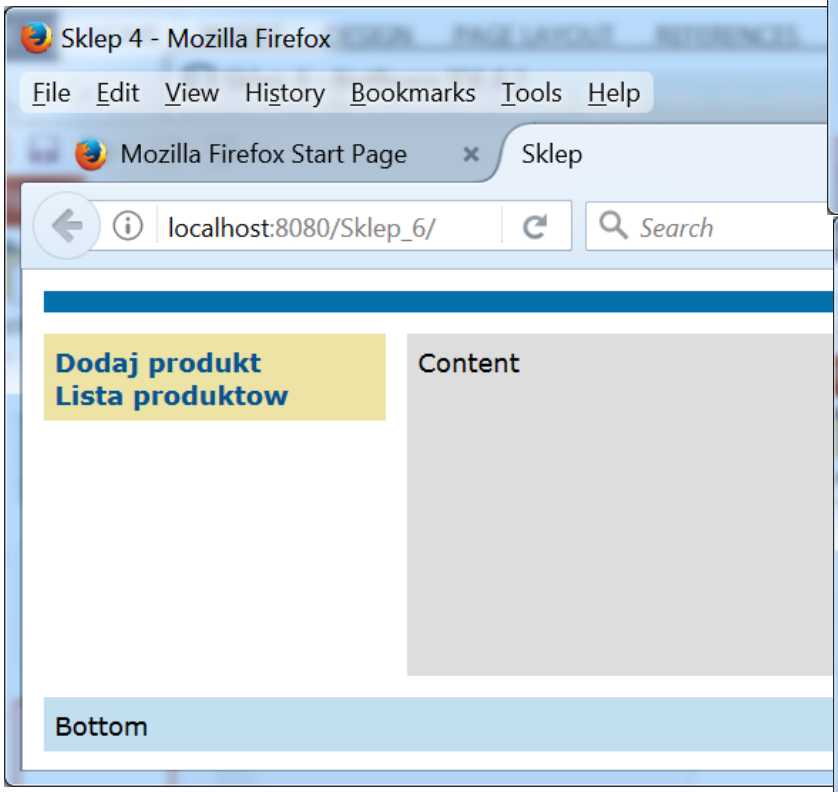


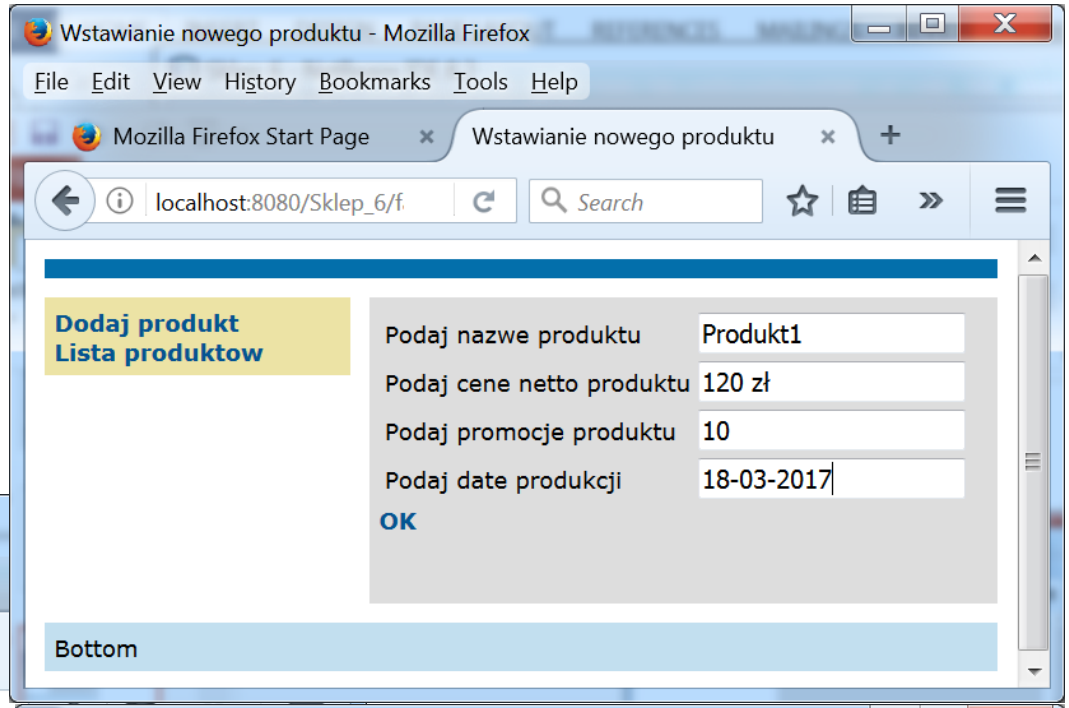
Po uruchomieniu formularza **Lista produktow** w obu instancjach klientów aplikacji widać, że korzystają **z tej samej instancji** komponentu typu **Fasada_warstwy_biznesowej_ejb**, czyli tej samej instancji obiektu **Fasada_warstwy_biznesowej.**

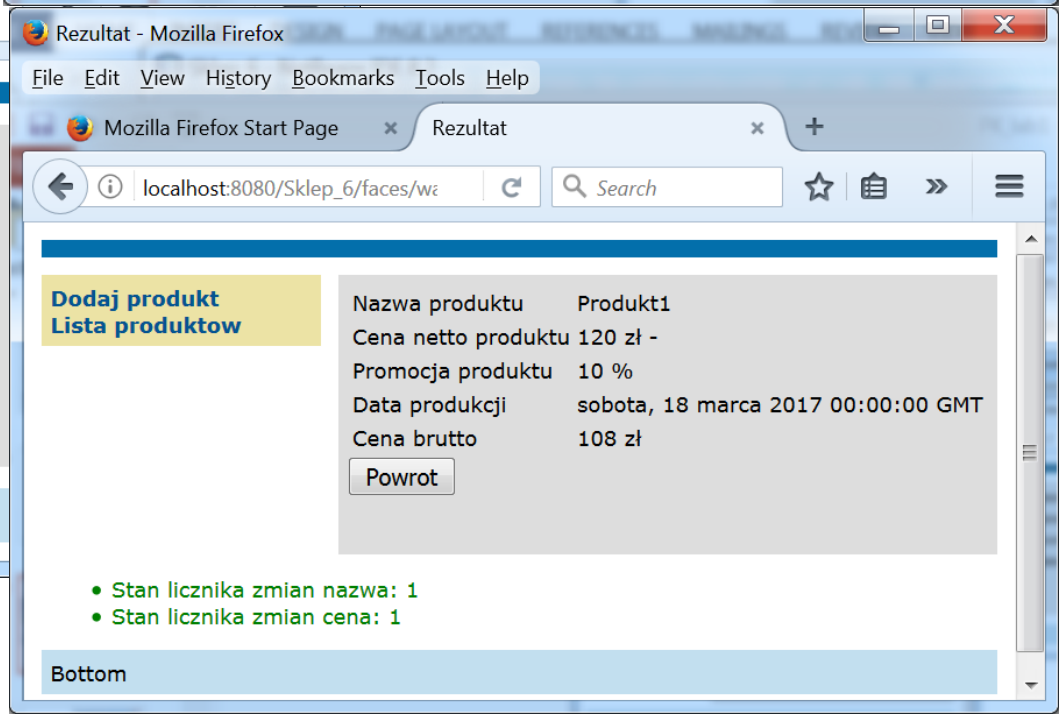


Uruchomienie aplikacji opartej na **komponencie EJB typu Session, rodzaj Stateless** - Clean and Build/Deploy/Run

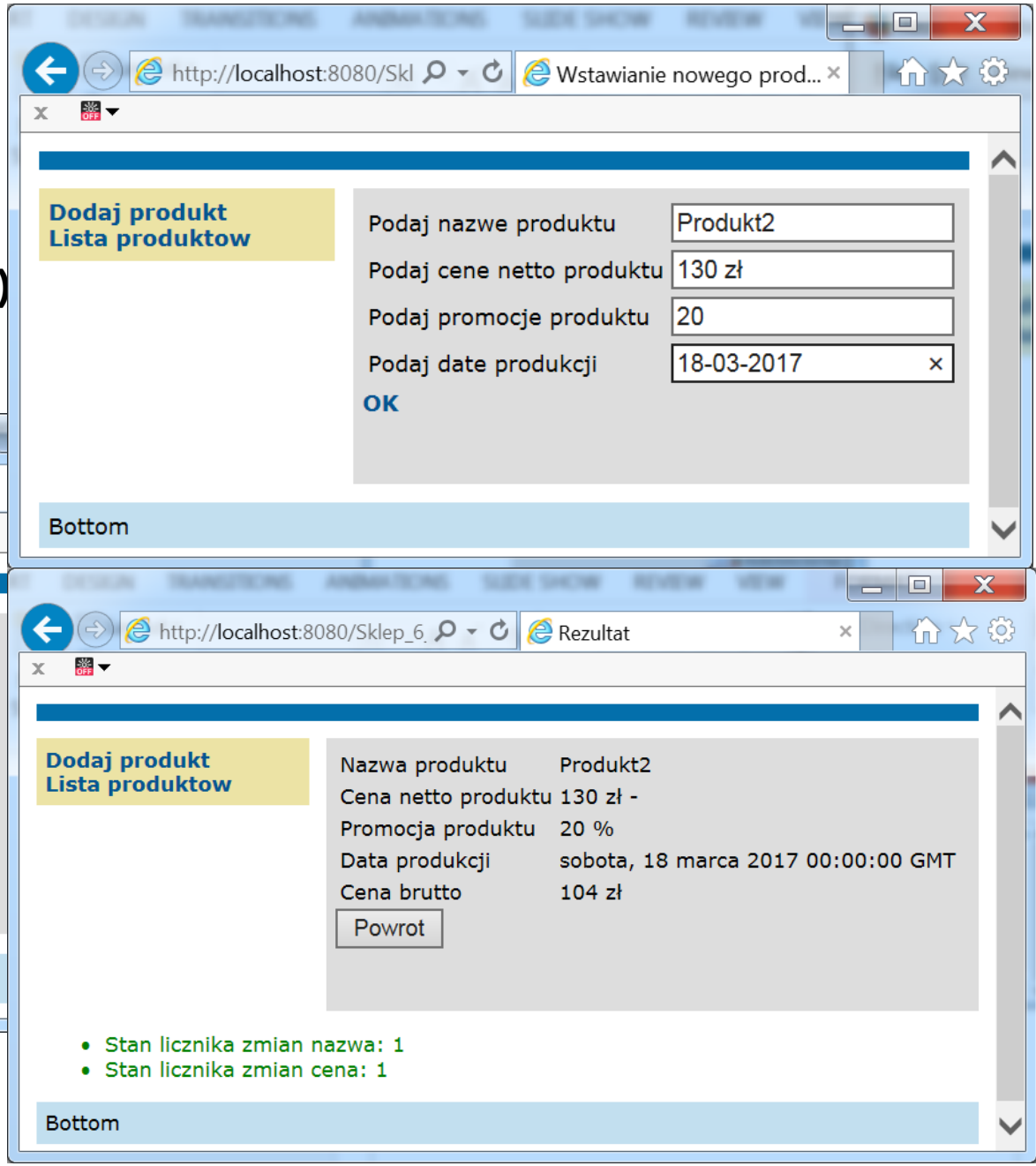
Uruchomienie 1-ej instancji warstwy klienta aplikacji i wprowadzenie danych produktu – może być uruchomiona automatycznie w przeglądarce domyślnej.

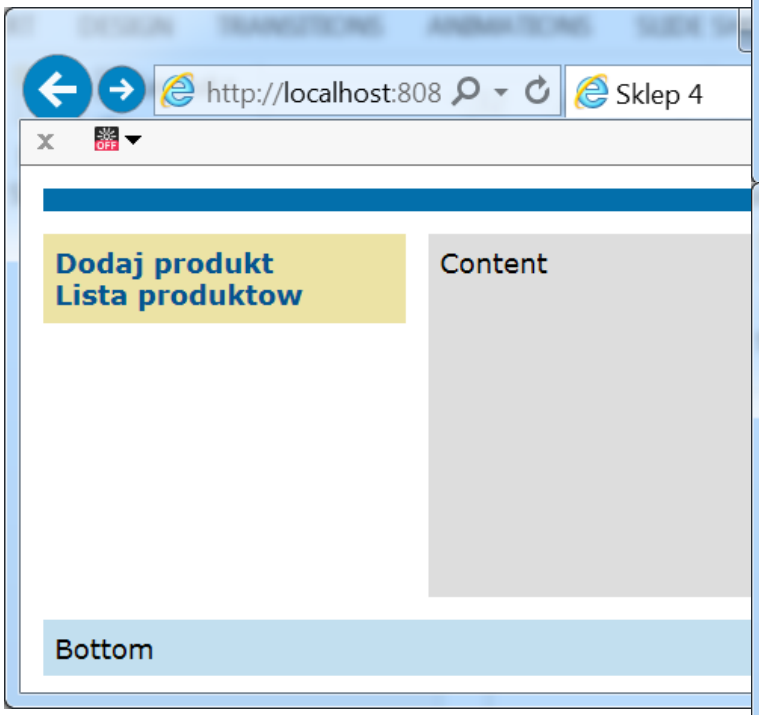




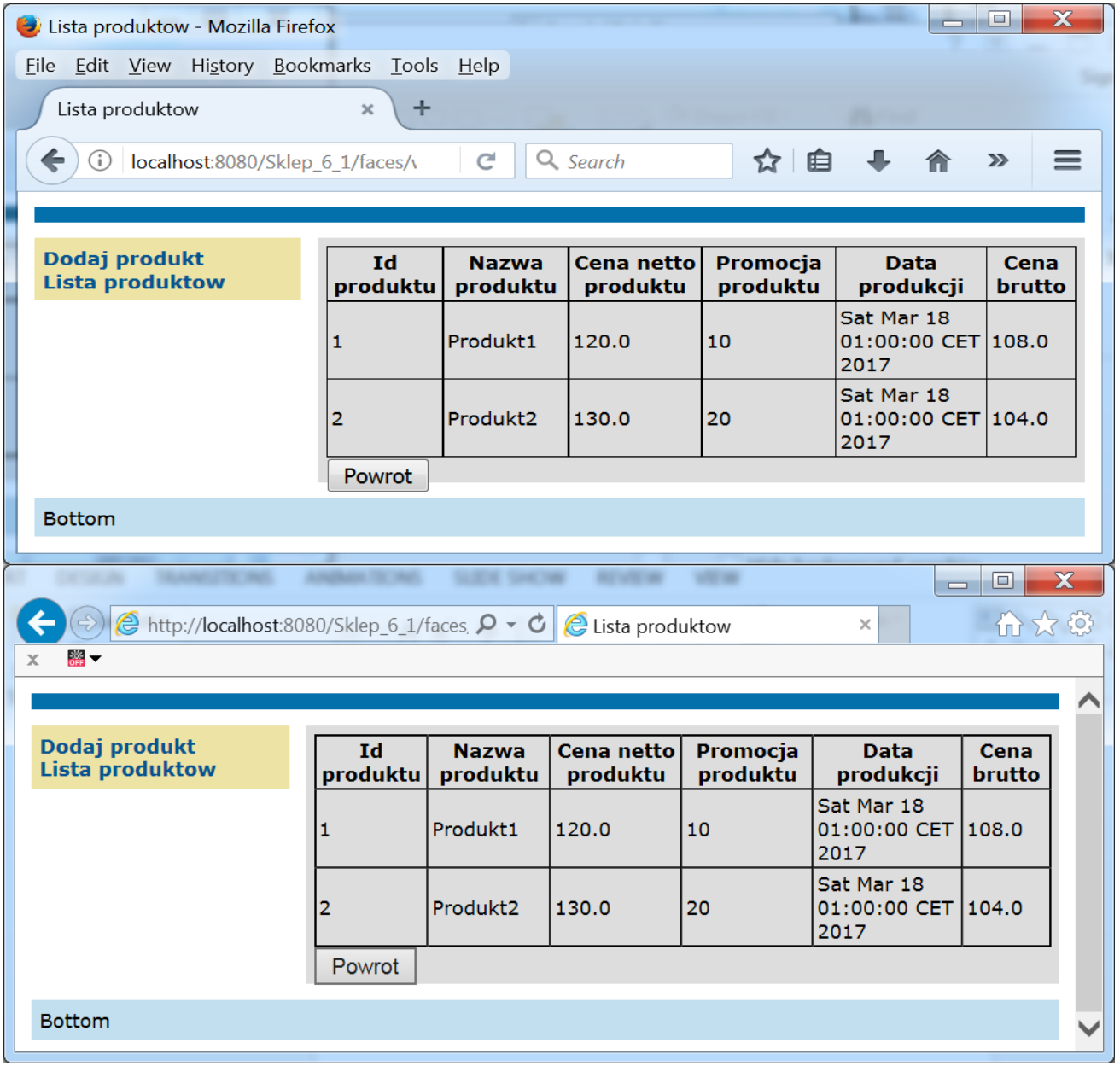


Uruchomienie 2-ej instancji warstwy klienta aplikacji aplikacji i wprowadzenie danych kolejnego produktu (prze wprowdzenie adresu: http://localhost:8080/Sklep_6_1)

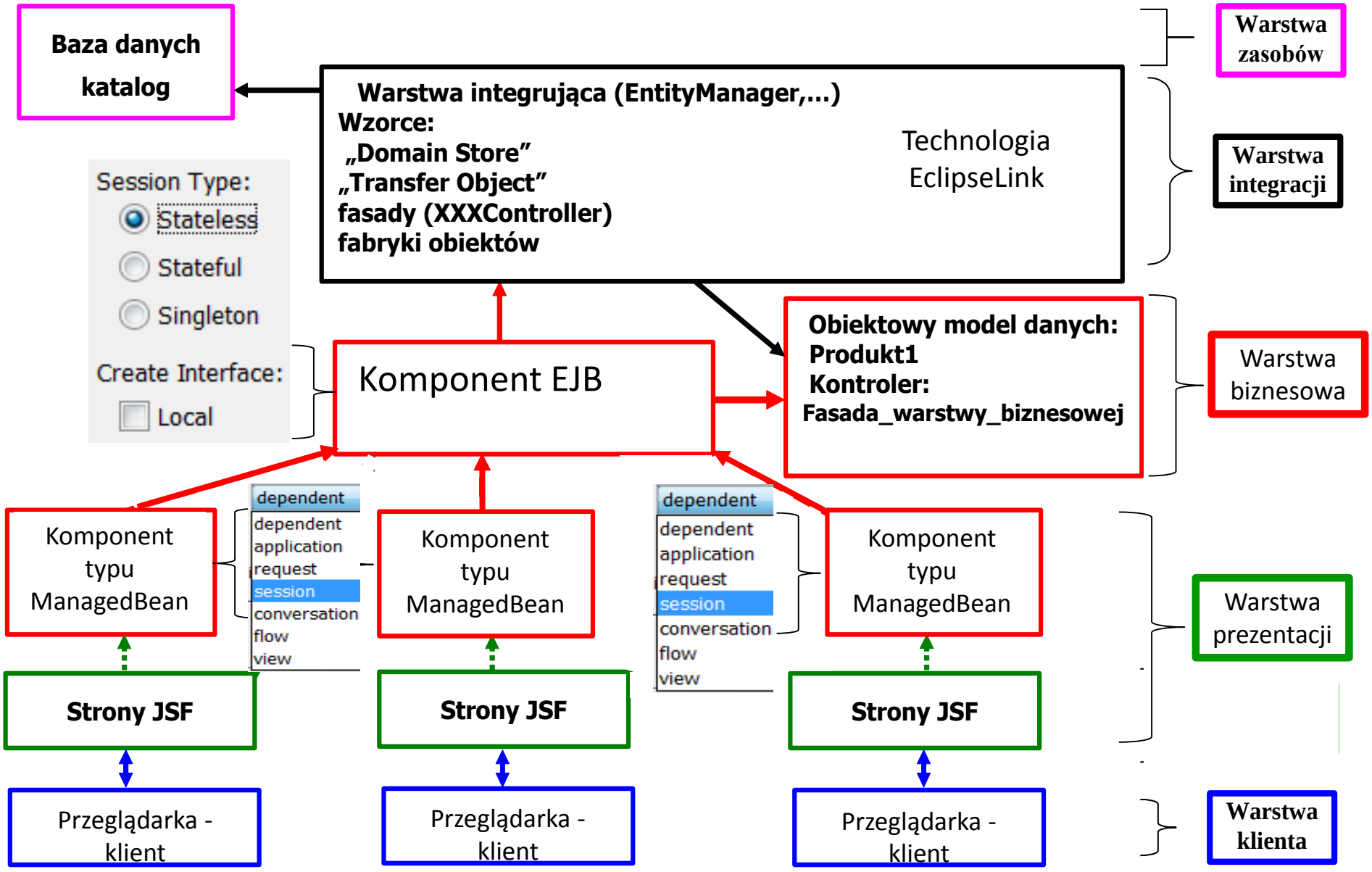




Po uruchomieniu formularza **Lista produktow** w obu instancjach klientów aplikacji widać, że korzystają **z tej samej instancji** komponentu typu **Fasada_warstwy_biznesowej_ejb**, czyli tej samej instancji obiektu **Fasada_warstwy_biznesowej.**

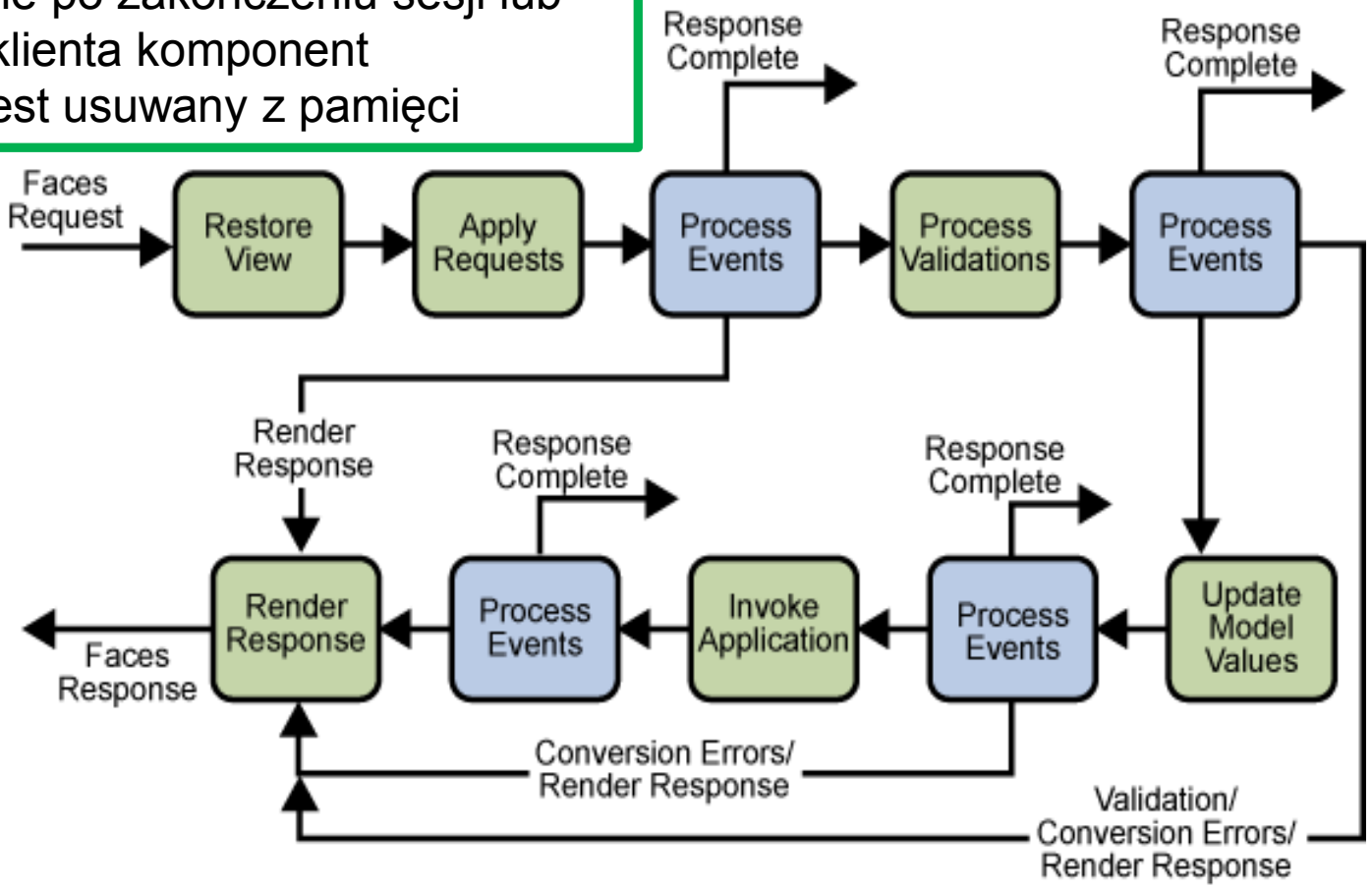


Ad. 1.2. Architektura aplikacji pięciowarstwowej -Java EE 7.0 JavaServer Faces



Standard cyklu życia „Request-Response” dla JavaServer Faces

SessionScoped: podczas uruchomienia instancji aplikacji klienta tworzony jest nowy komponent typu **Managed_produkt** – wszystkie jego dane są zaktualizowane podczas fazy Invoke Application. Po zakończeniu fazy Render Response komponent pozostaje w pamięci. Jeśli nie wystąpią błędy, jedynie po zakończeniu sesji lub zamknięciu aplikacji klienta komponent **Managed_produkt** jest usuwany z pamięci



Tworzenie modelu komponentu dataTable w komponencie typu Managed_produkt na stronie lista_produktow.xhtml

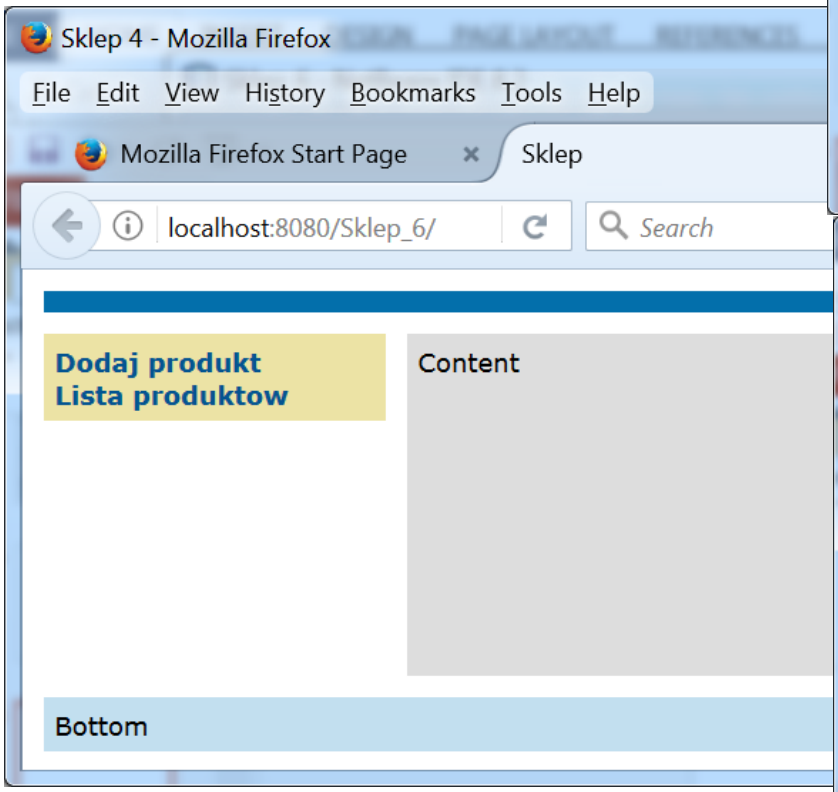
```
public DataModel utworz_DataModel() {
    return new ListDataModel(fasada.items());
}

public DataModel getItems() {
    if (items == null || stan==1) {
        items = utworz_DataModel();
}
    return items;
}
```

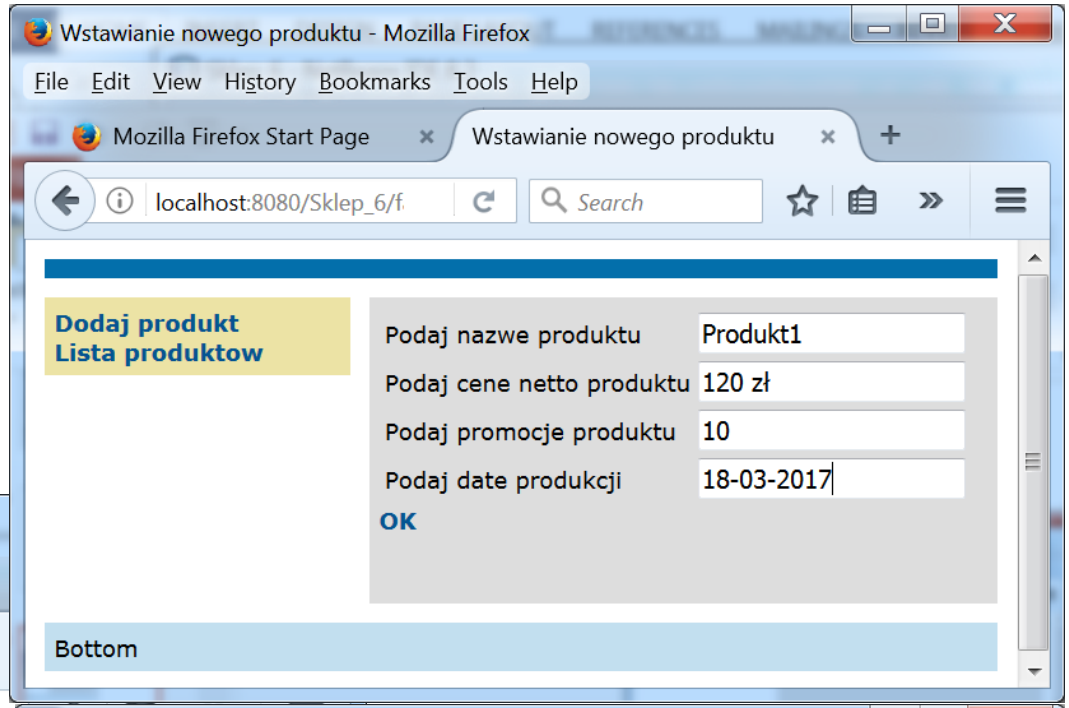
Przy czasie życia SessionScope obiektu typu Managed_produkt atrybut items jest **równy null** tylko podczas obsługi pierwszego żądania. Badanie zmiennej stan pozwala na aktualizaję modelu items: stan==1 oznacza, **że dodano nowe dane w danej instancji aplikacji internetowej.**

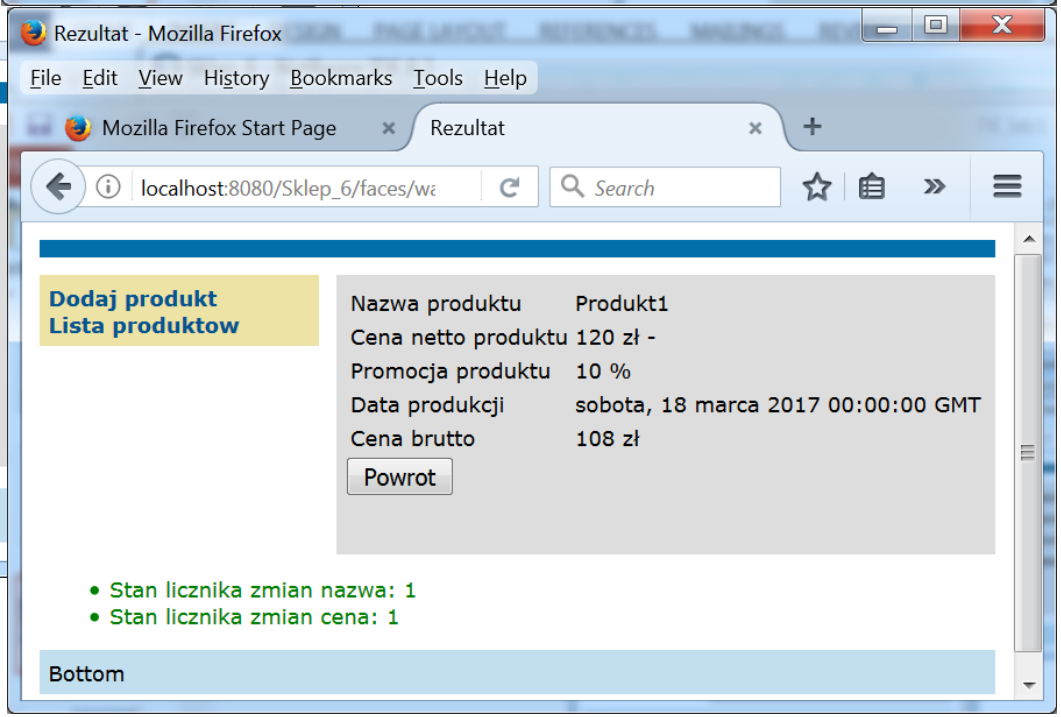
Model elemetów strony **rezultat2.xhtml** jest tworzony zawsze – albo pokazane są dane nowego produktu lub pojawia się napis o braku wstawienia nowego produktu (zmienna stan=0)

Uruchomienie aplikacji opartej na **komponencie EJB typu Session, rodzaj Stateless** - Clean and Build/Deploy/Run

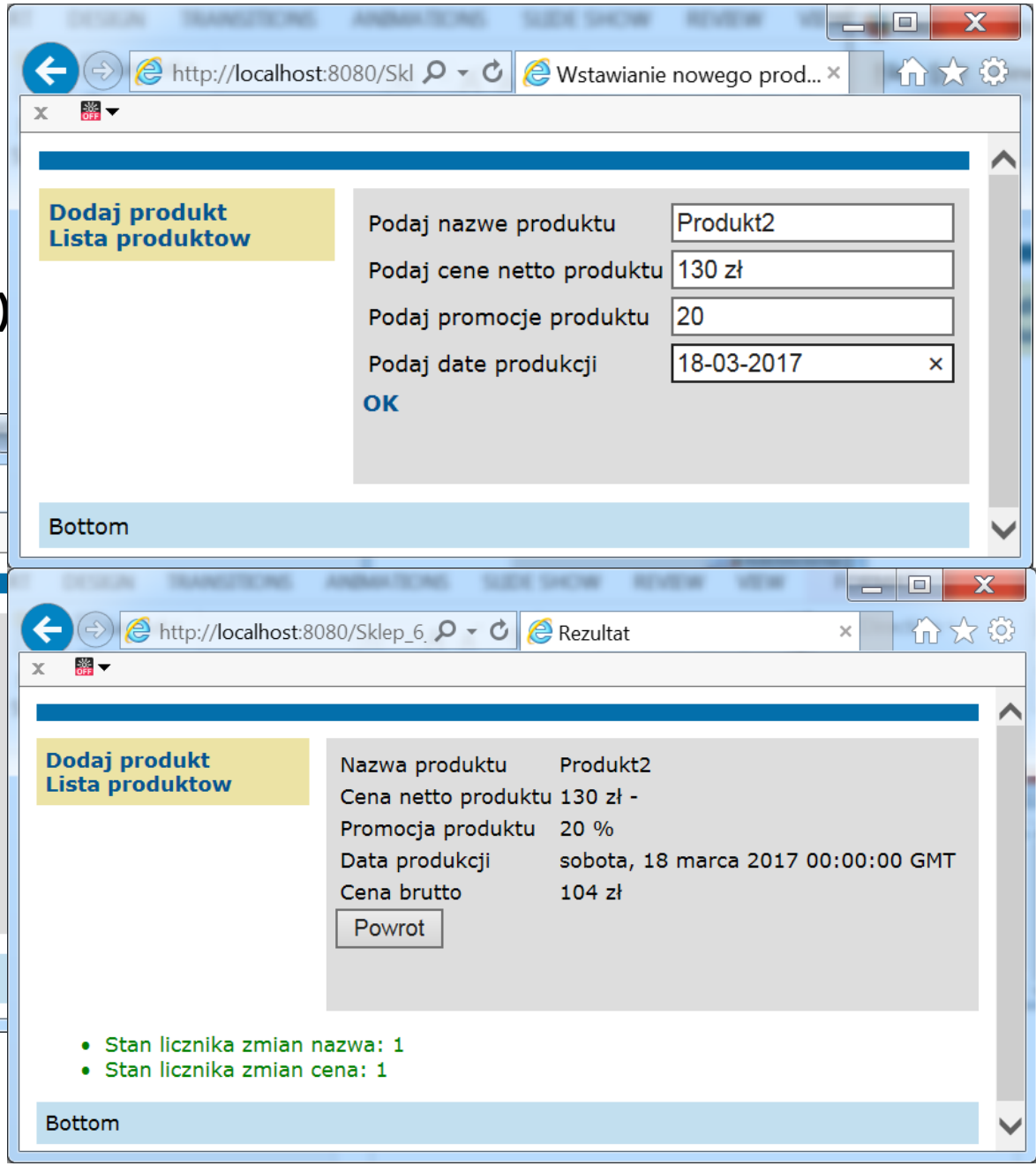
Uruchomienie 1-ej instancji warstwy klienta aplikacji i wprowadzenie danych produktu – może być uruchomiona automatycznie w przeglądarce domyślnej.

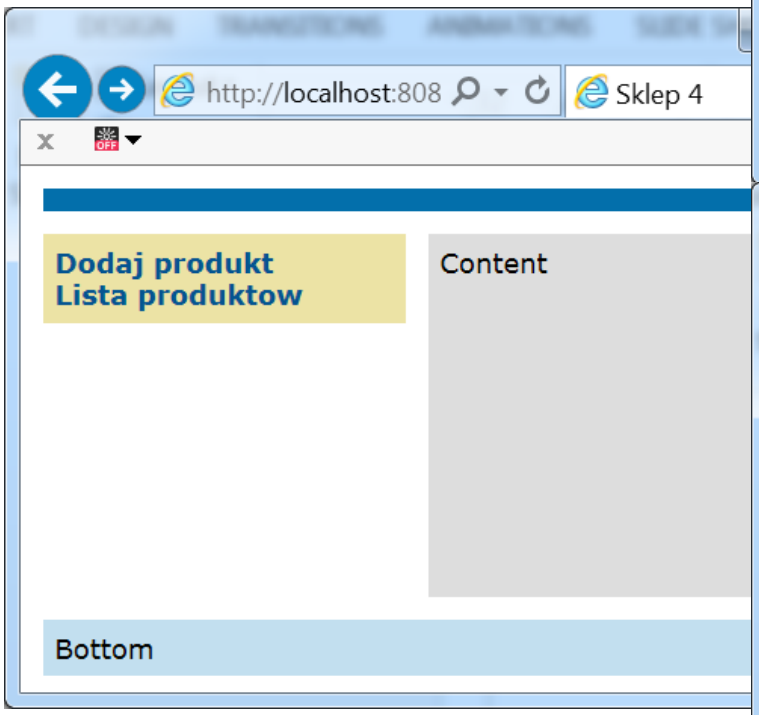




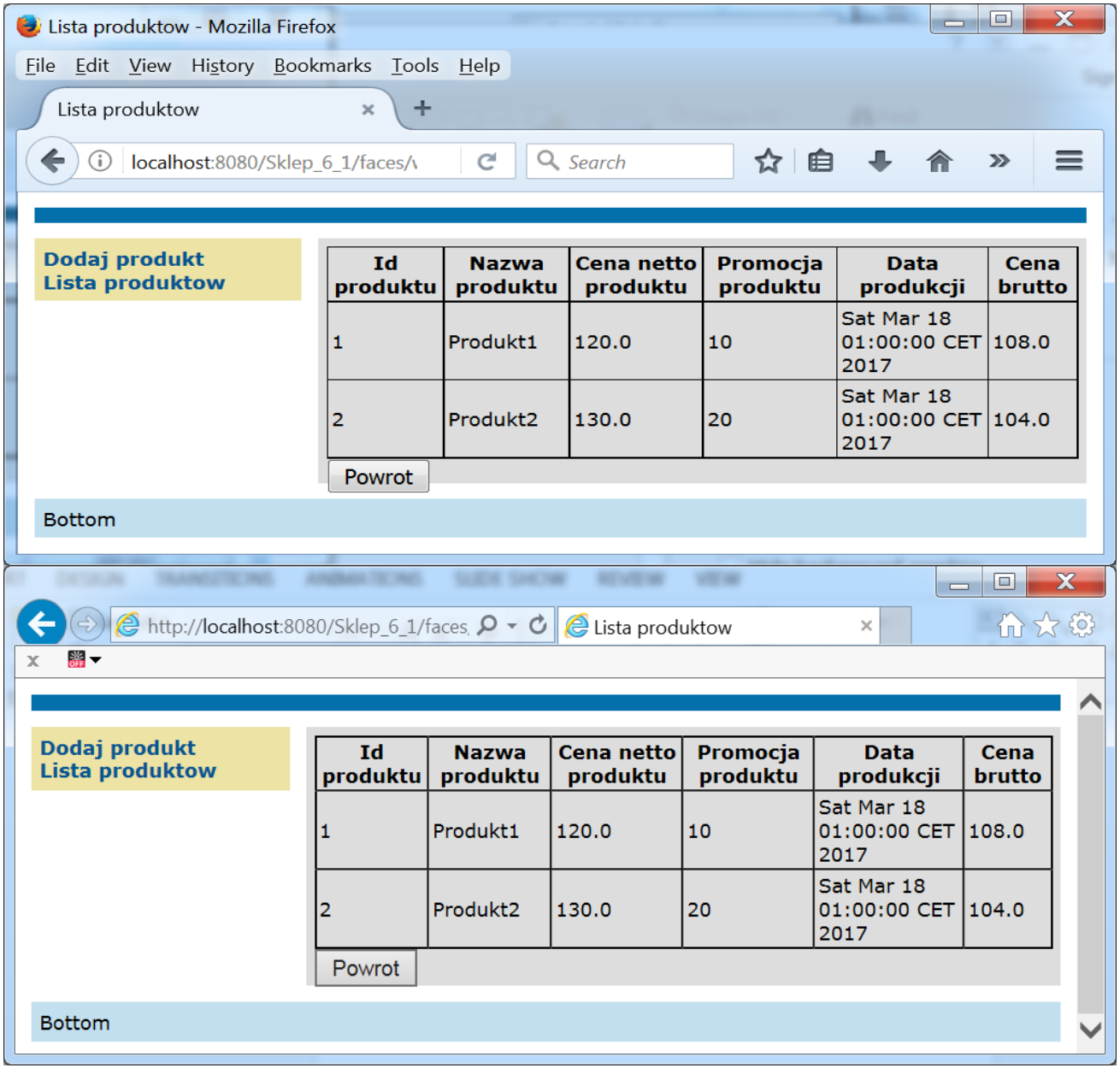


Uruchomienie 2-ej instancji warstwy klienta aplikacji aplikacji i wprowadzenie danych kolejnego produktu (przez wprowdzenie adresu: http://localhost:8080/Sklep_6_1)



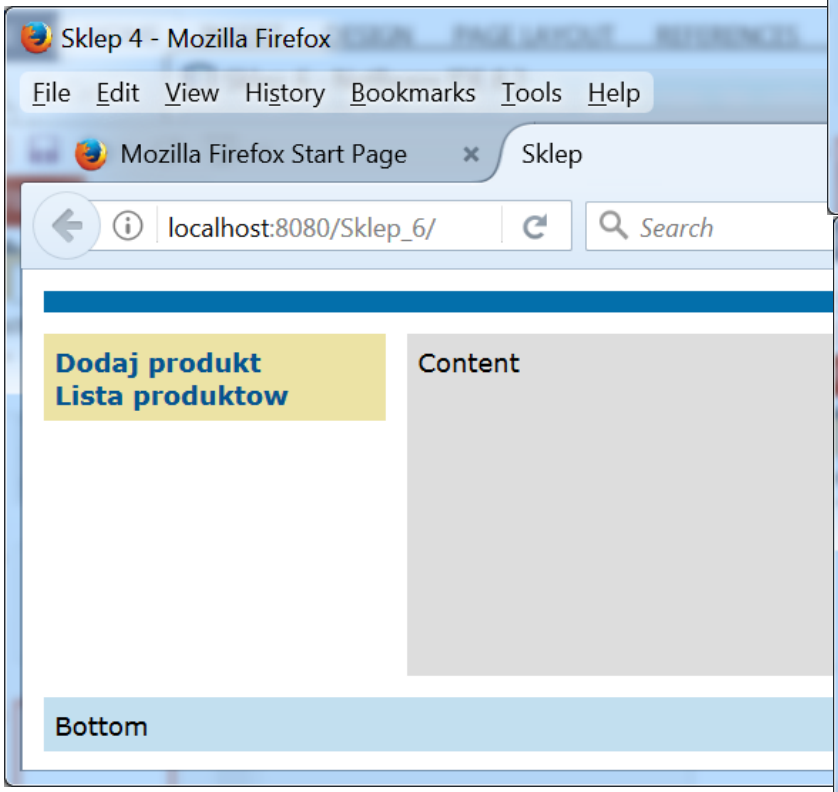


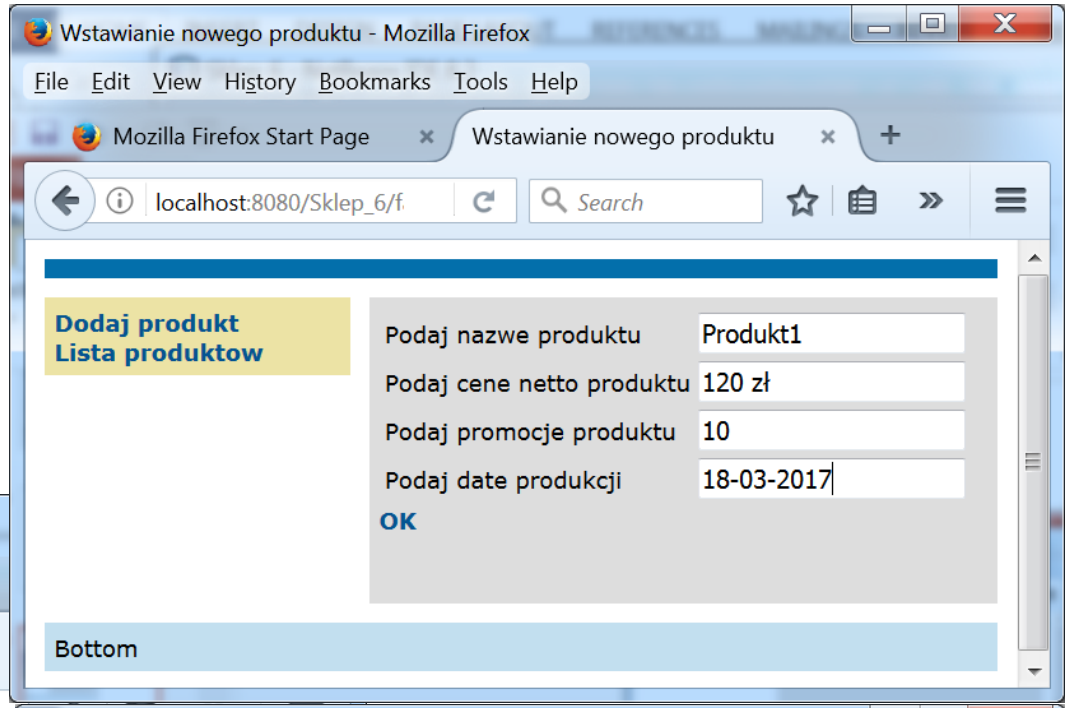
Po uruchomieniu formularza **Lista produktow** w obu instancjach klientów aplikacji widać, że korzystają **z tej samej instancji** komponentu typu **Fasada_warstwy_biznesowej_ejb**, czyli tej samej instancji obiektu **Fasada_warstwy_biznesowej.**

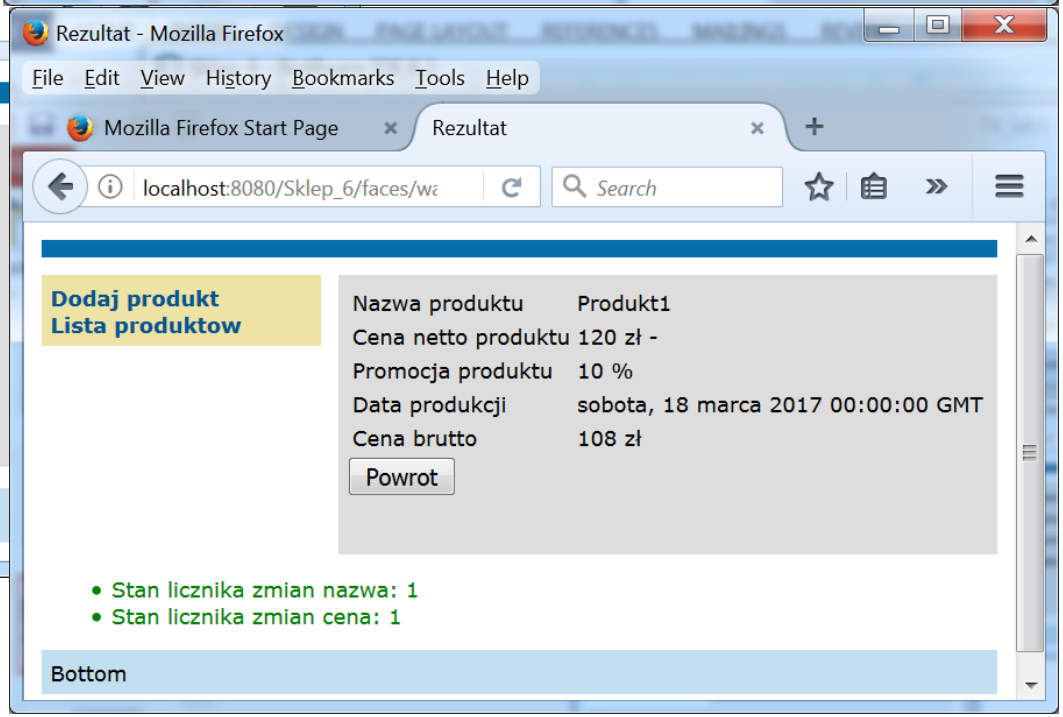


Uruchomienie aplikacji opartej na **komponencie EJB typu Session, rodzaj Stateless** - Clean and Build/Deploy/Run

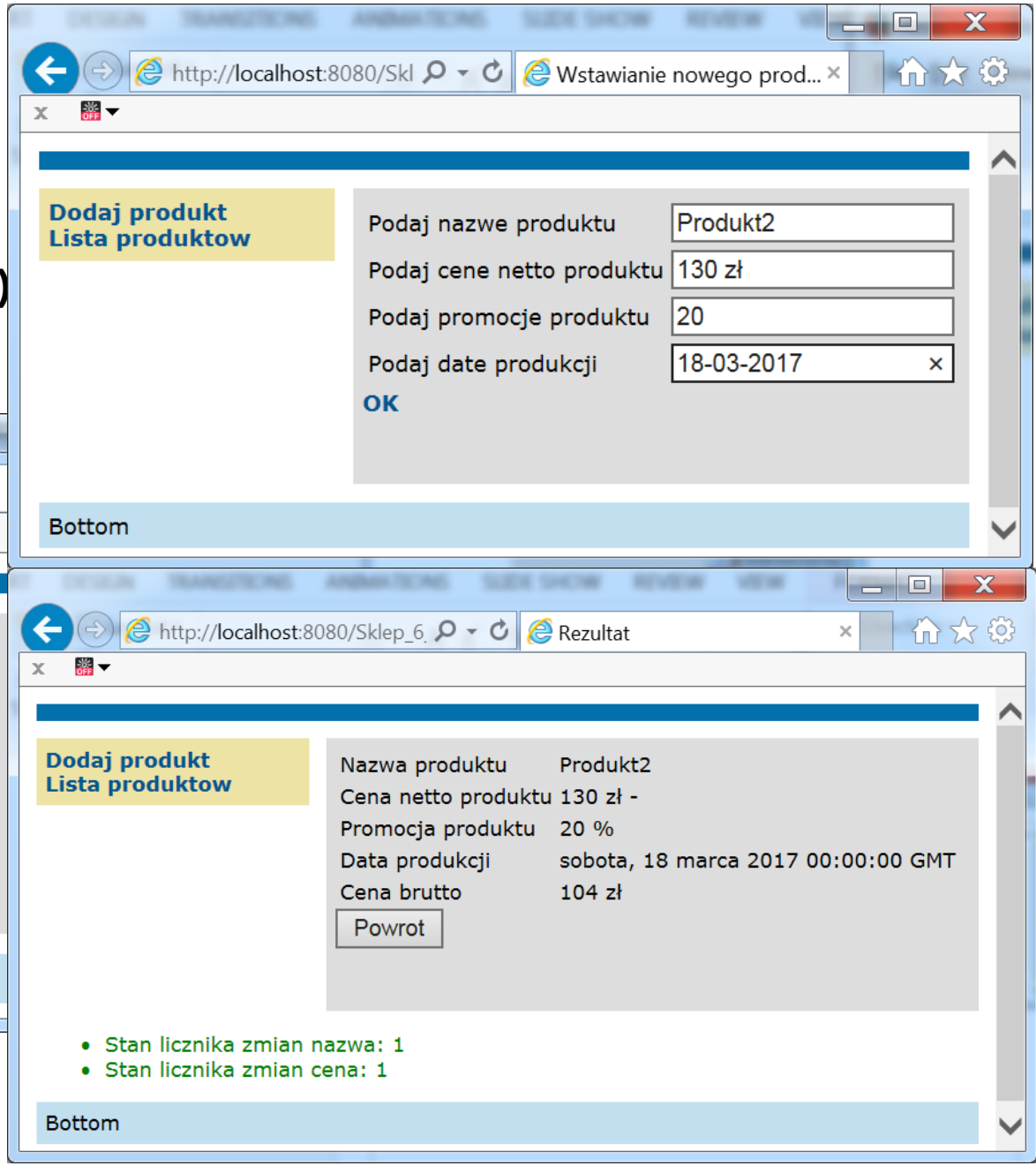
Uruchomienie 1-ej instancji warstwy klienta aplikacji i wprowadzenie danych produktu – może być uruchomiona automatycznie w przeglądarce domyślnej.

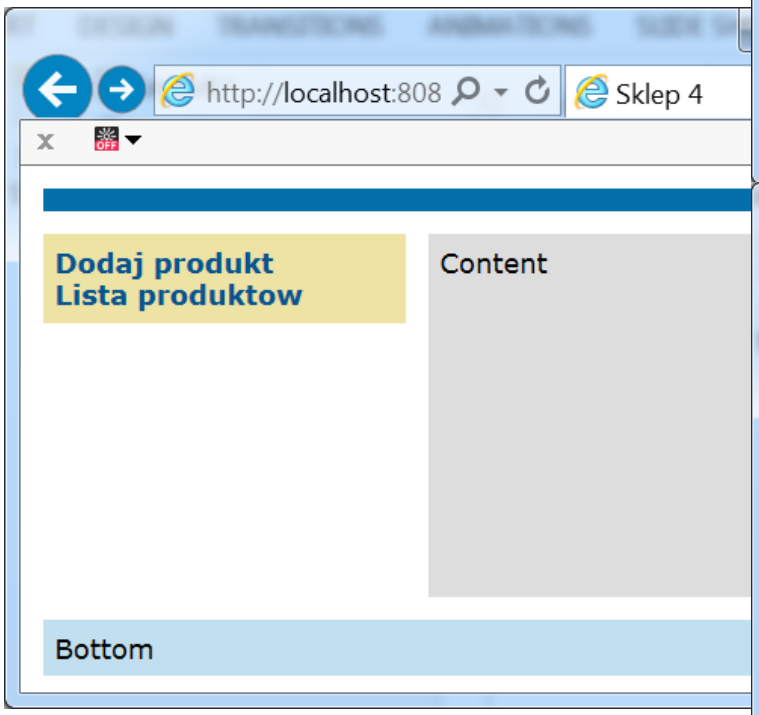




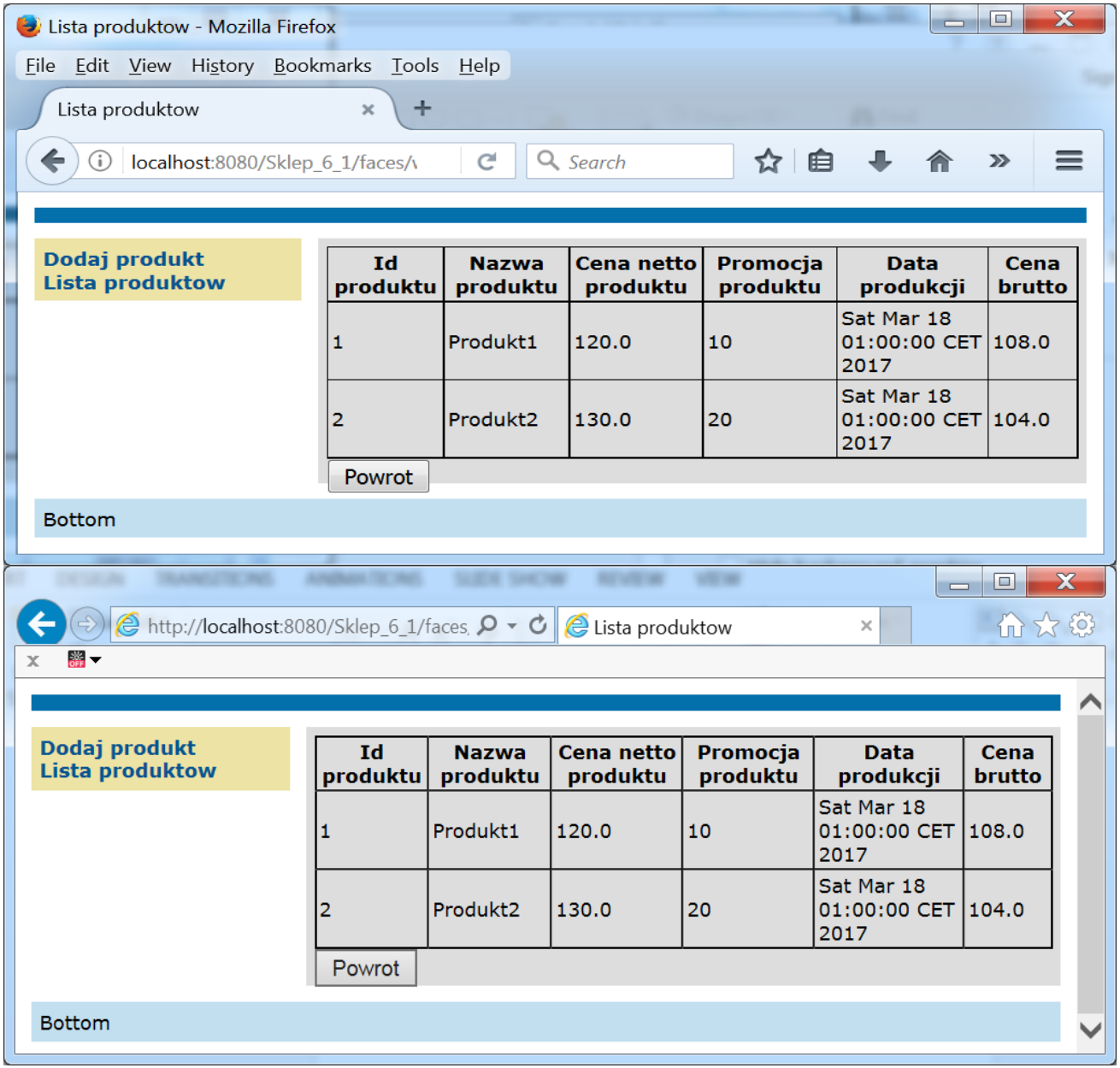


Uruchomienie 2-ej instancji warstwy klienta aplikacji aplikacji i wprowadzenie danych kolejnego produktu (przez wprowdzenie adresu: http://localhost:8080/Sklep_6_1)





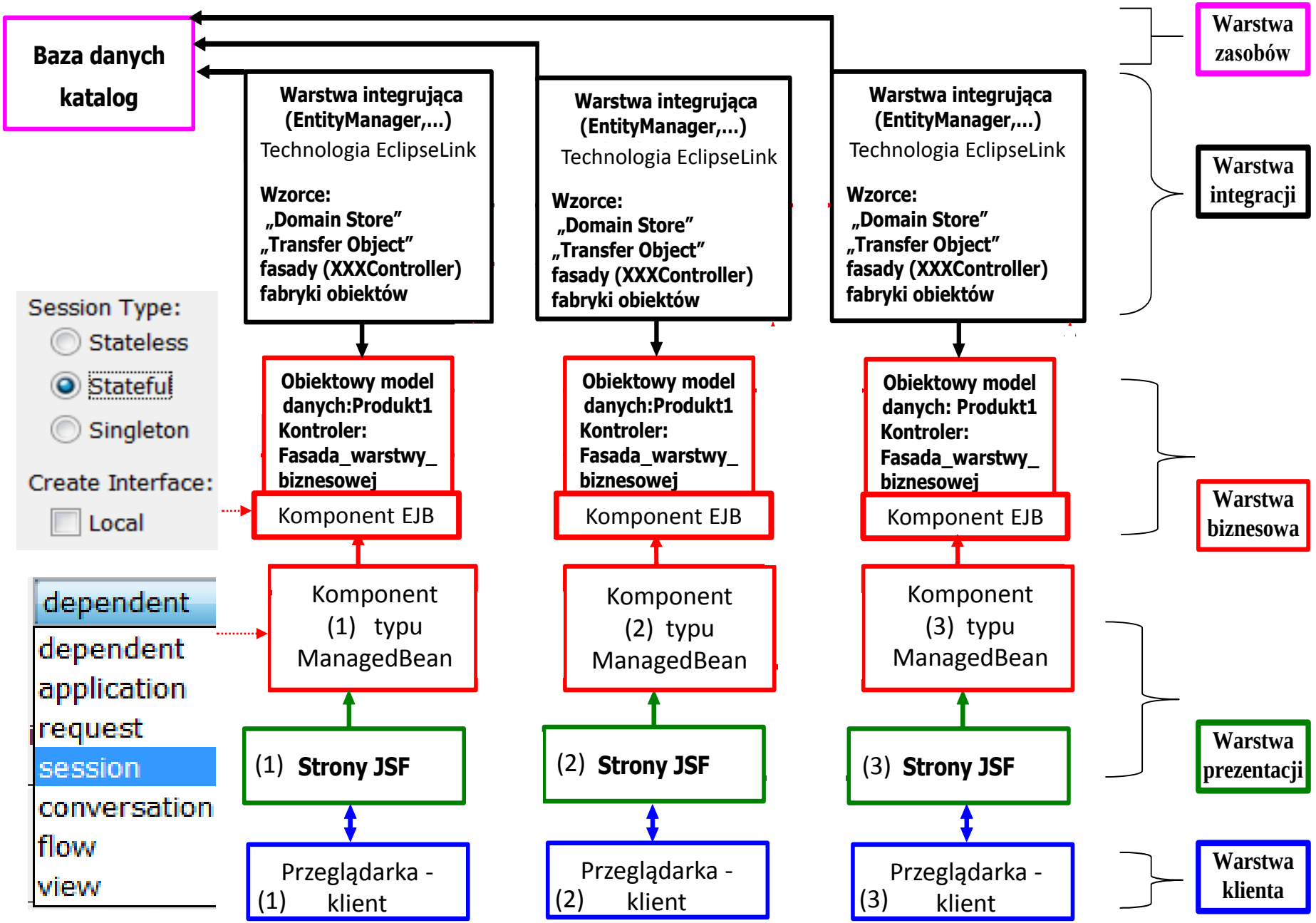
Po uruchomieniu formularza **Lista produktow** w obu instancjach klientów aplikacji widać, że korzystają **z tej samej instancji** komponentu typu **Fasada_warstwy_biznesowej_ejb**, czyli tej samej instancji obiektu **Fasada_warstwy_biznesowej.**



2. Aplikacja wielowarstwowa Java EE oparta na komponencie EJB typu Session Bean – rodzaj Stateful

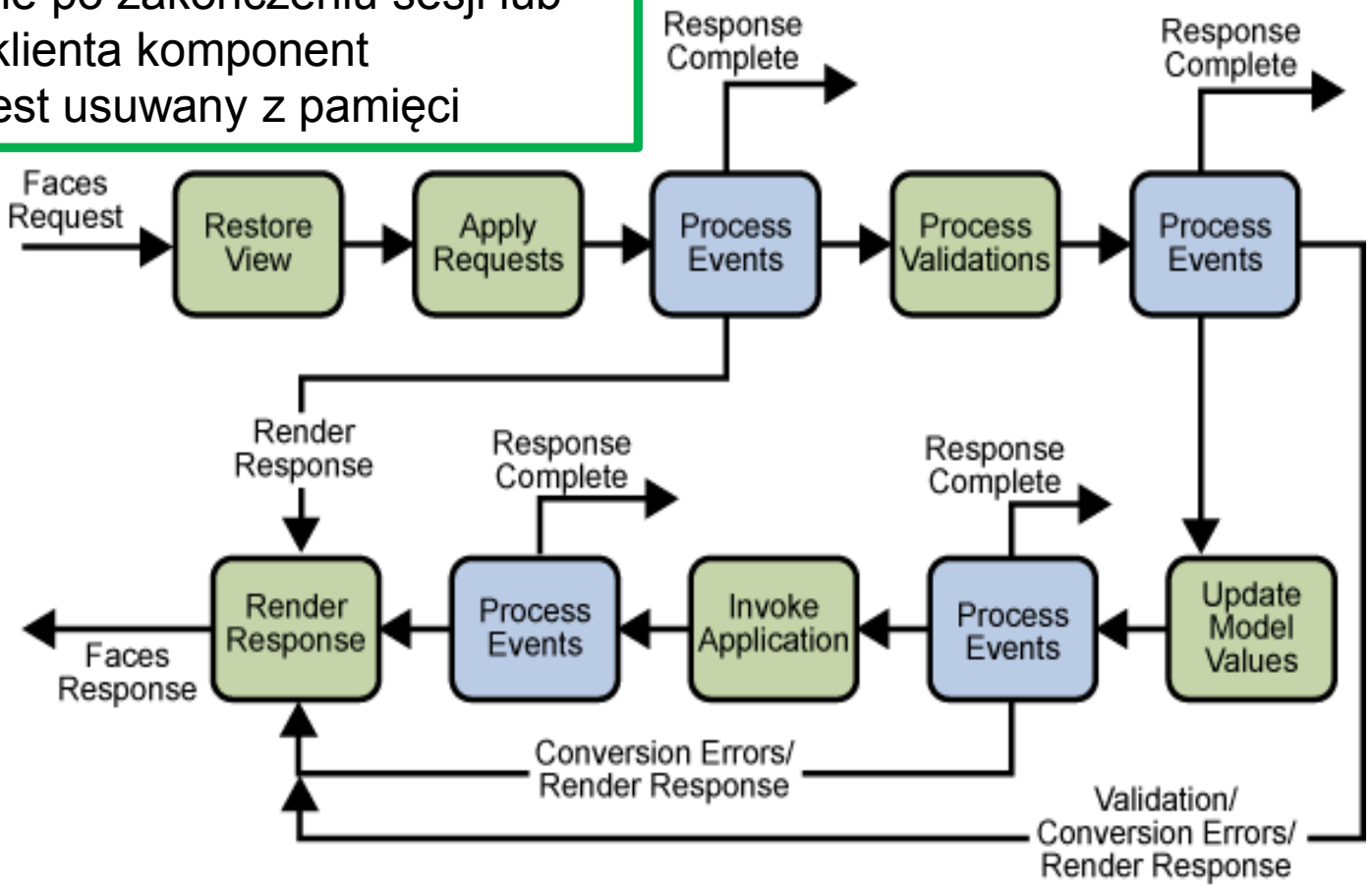
Warstwa internetowa oparta na komponencie Managed Bean o czasie życia **SessionScope**

Architektura aplikacji pięciowarstwowej – Java EE 7.0 JavaServer Faces



Standard cyklu życia „Request-Response” dla JavaServer Faces

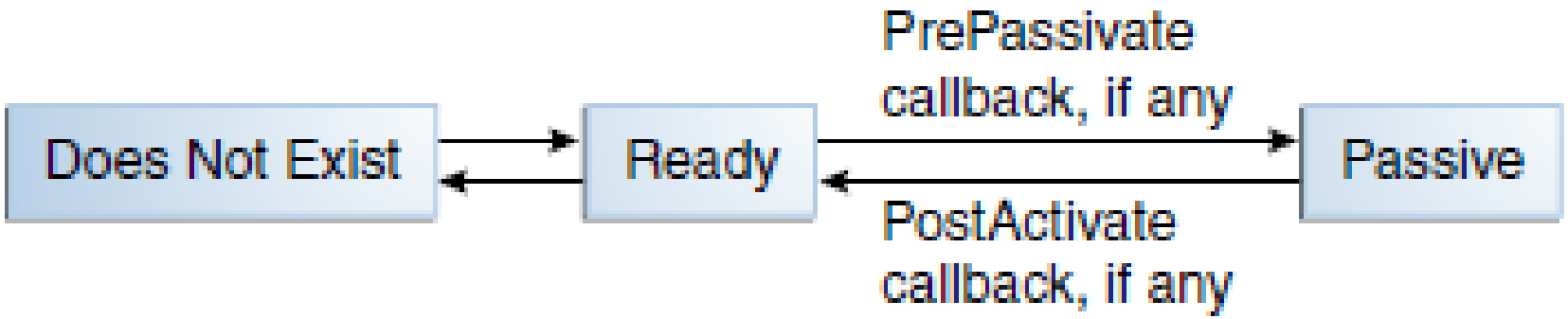
SessionScoped: podczas uruchomienia instancji aplikacji klienta tworzony jest nowy komponent typu **Managed_produkt** – wszystkie jego dane są zaktualizowane podczas fazy Invoke Application. Po zakończeniu fazy Render Response komponent pozostaje w pamięci. Jeśli nie wystąpią błędy, jedynie po zakończeniu sesji lub zamknięciu aplikacji klienta komponent **Managed_produkt** jest usuwany z pamięci



Cykl życia stanowych (Stateful) komponentów typu Session Bean

Zadania kontenera EJB:

1. Tworzy komponent
2. Dependency injection, jeśli są
3. Metoda PostConstruct callback, jeśli jest
4. Metoda Init, lub ejbCreate<METHOD>, jeśli są



1. Usuwa komponent
2. Metoda PreDestroy callback, jeśli jest

Tworzenie modelu komponentu dataTable w kompoenencie typu Managed_produkt na stronie lista_produktow.xhtml – podobnie jak w p.1.2

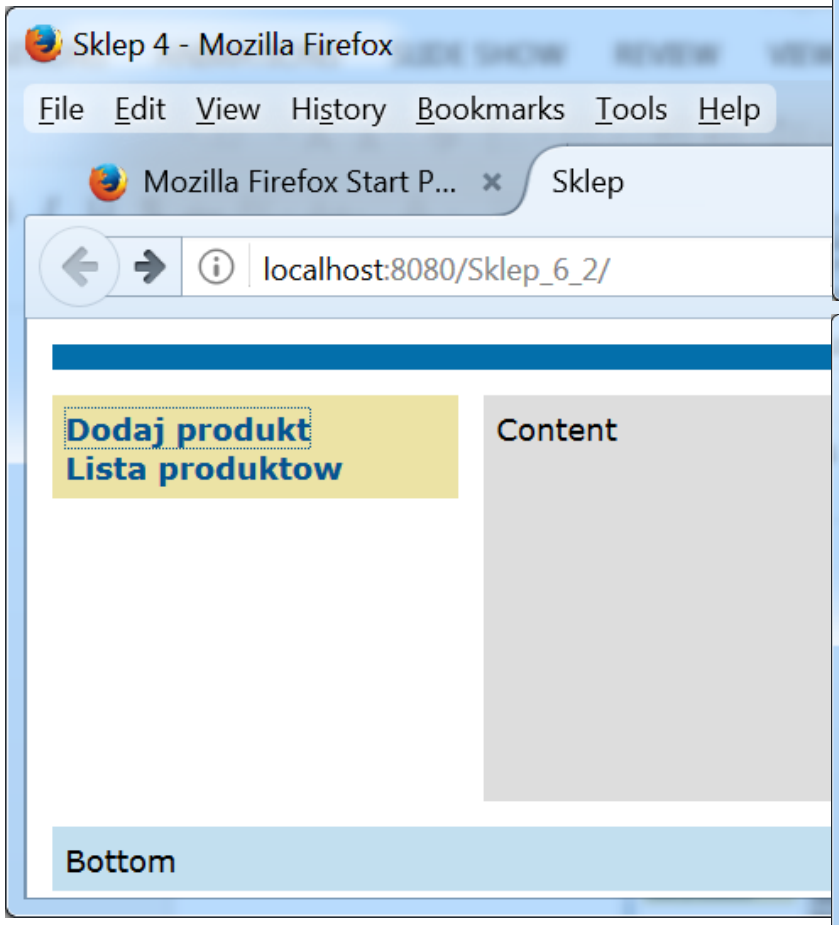
```
public DataModel utworz_DataModel() {
      return new ListDataModel(fasada.items());
  }

  public DataModel getItems() {
      if (items == null || stan==1) {
          items = utworz_DataModel();
          stan=0;
       }
      return items;
  }
```

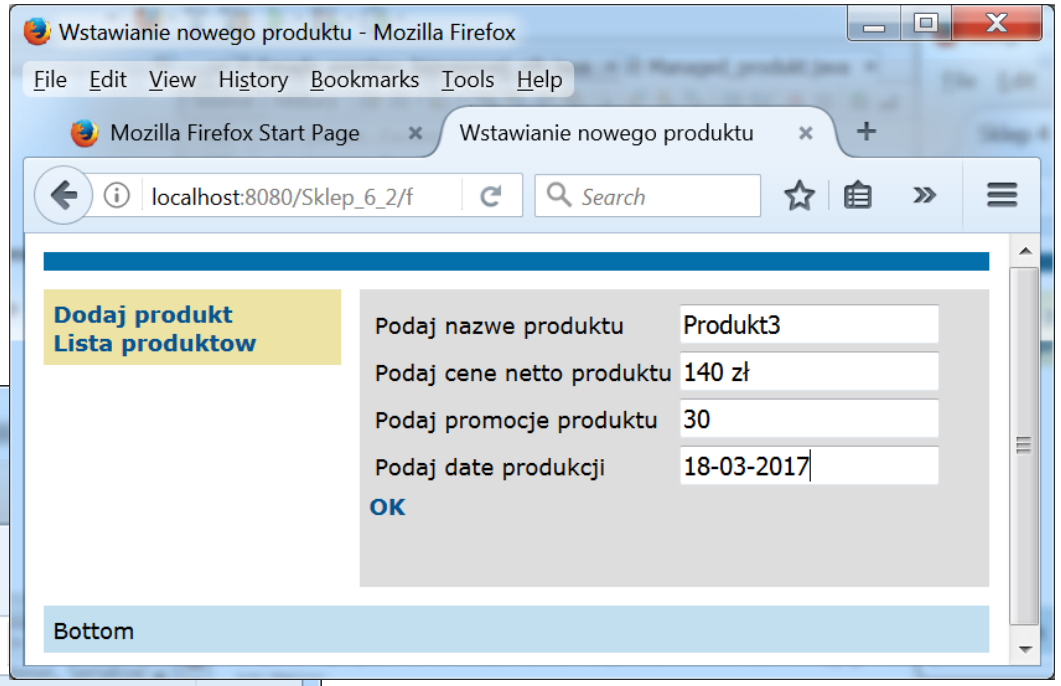
Przy czasie życia SessionScope obiektu typu Managed_produkt items jest **równy null** tylko podczas obsługi pierwszego żądania pierwszego. Badanie zmiennej stan pozwala na właściwą aktualizaję modelu items: stan==1 oznacza, że **dodano nowe dane w danej instancji klienta internetowego**

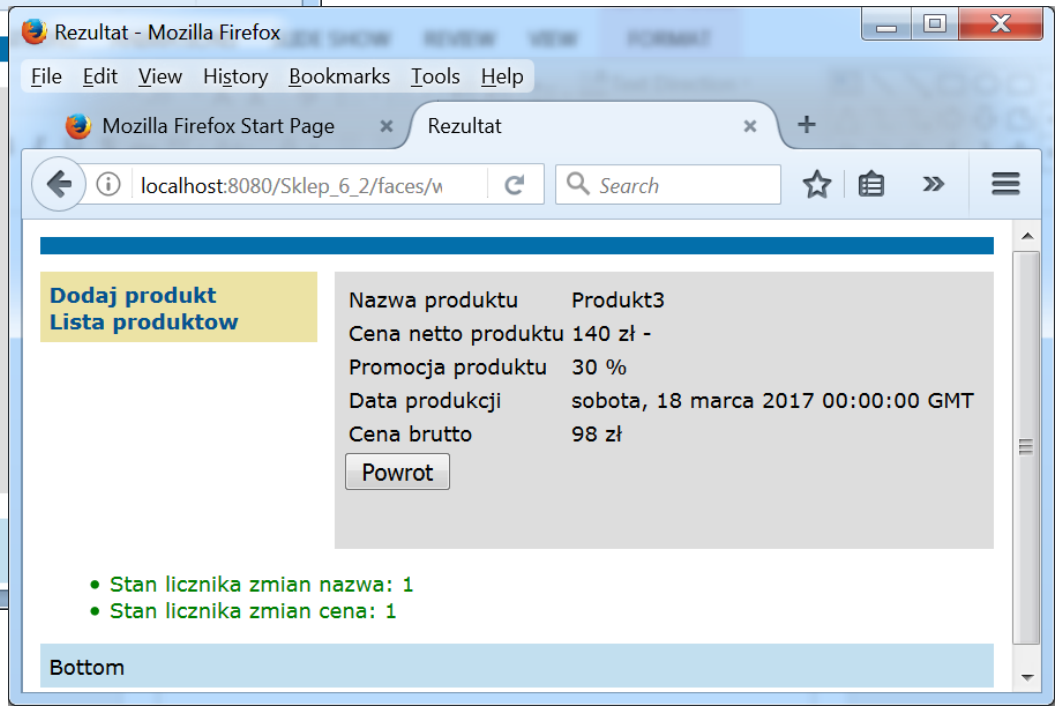
Model elemetów strony **rezultat2.xhtml** jest tworzony zawsze – albo pokazane są dane nowego produktu lub pojawia się napis o braku wstawienia nowego produktu (zmienna stan=0)

Uruchomienie aplikacji opartej na **komponencie EJB typu Session, rodzaj Stateful** - Clean and Build/Deploy/Run

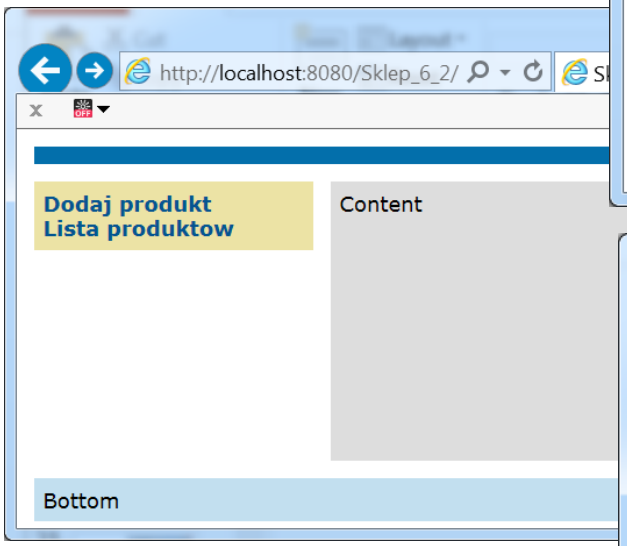
Uruchomienie 1-ej instancji warstwy klienta aplikacji i wprowadzenie danych produktu

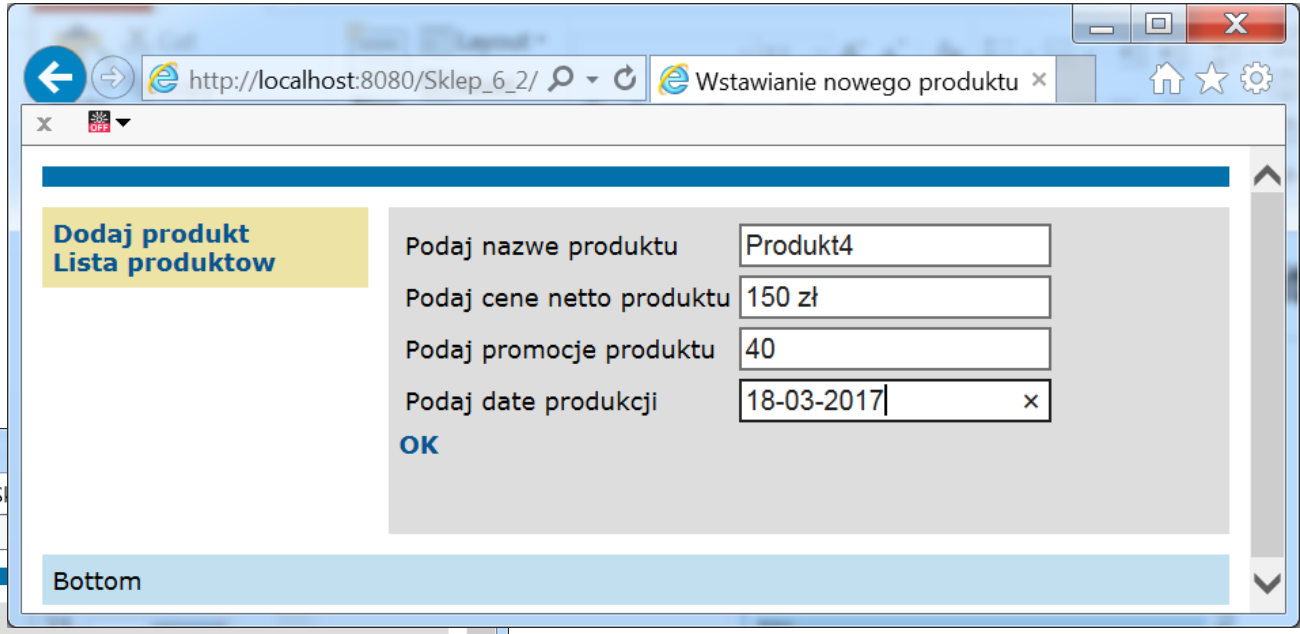


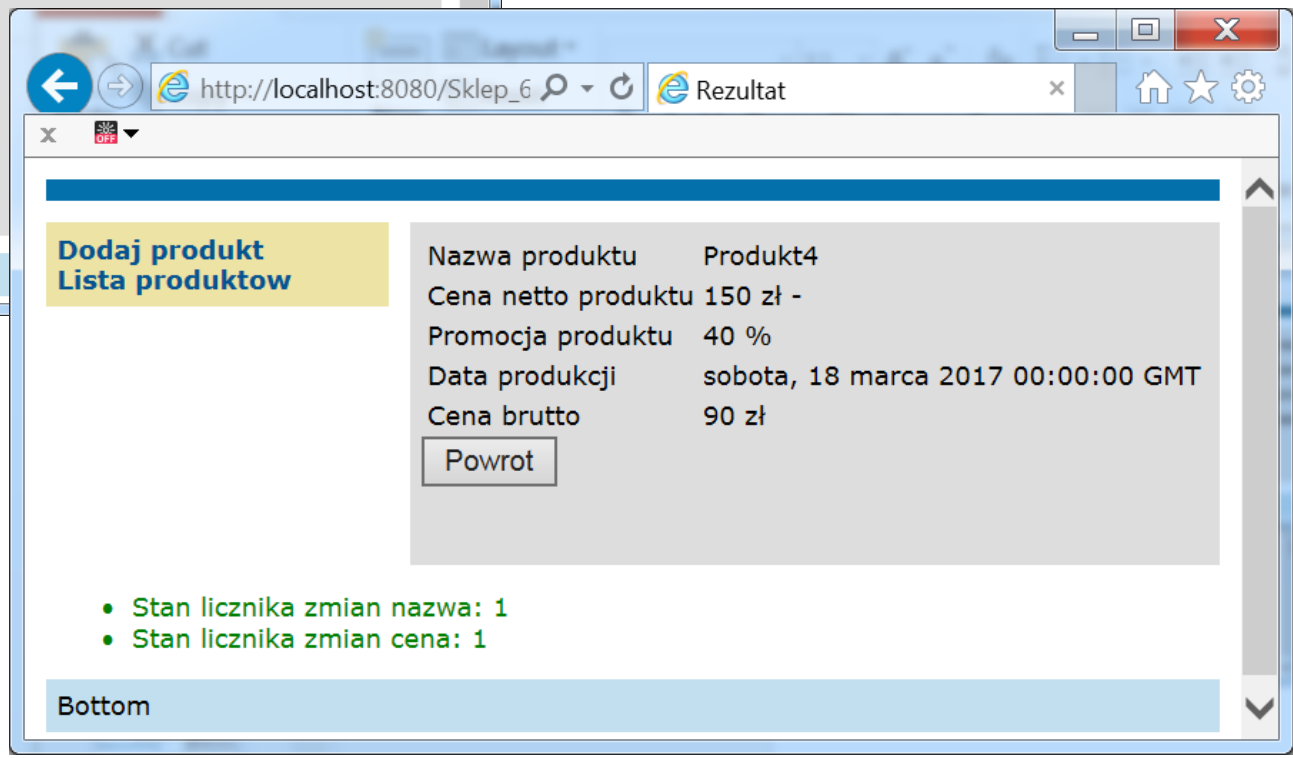




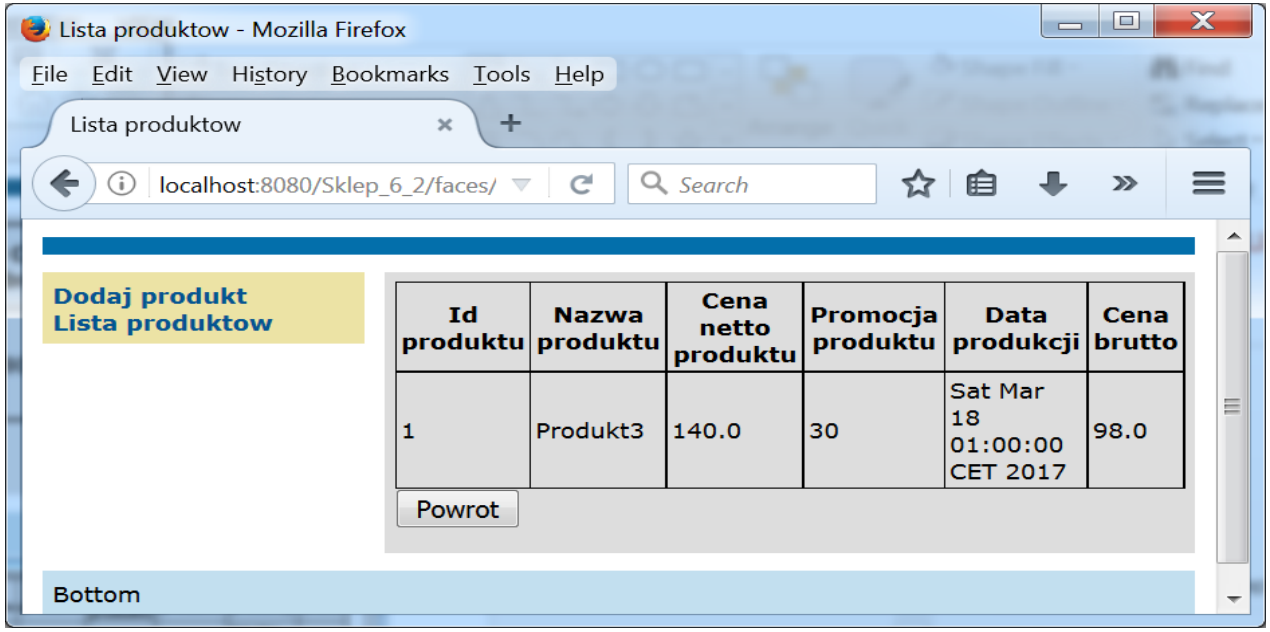
Uruchomienie 2-ej instancji warstwy klienta aplikacji i wprowadzenie danych produktu (podobnie jak w p.1.16: http://localhost:8080/Sklep_6_2)

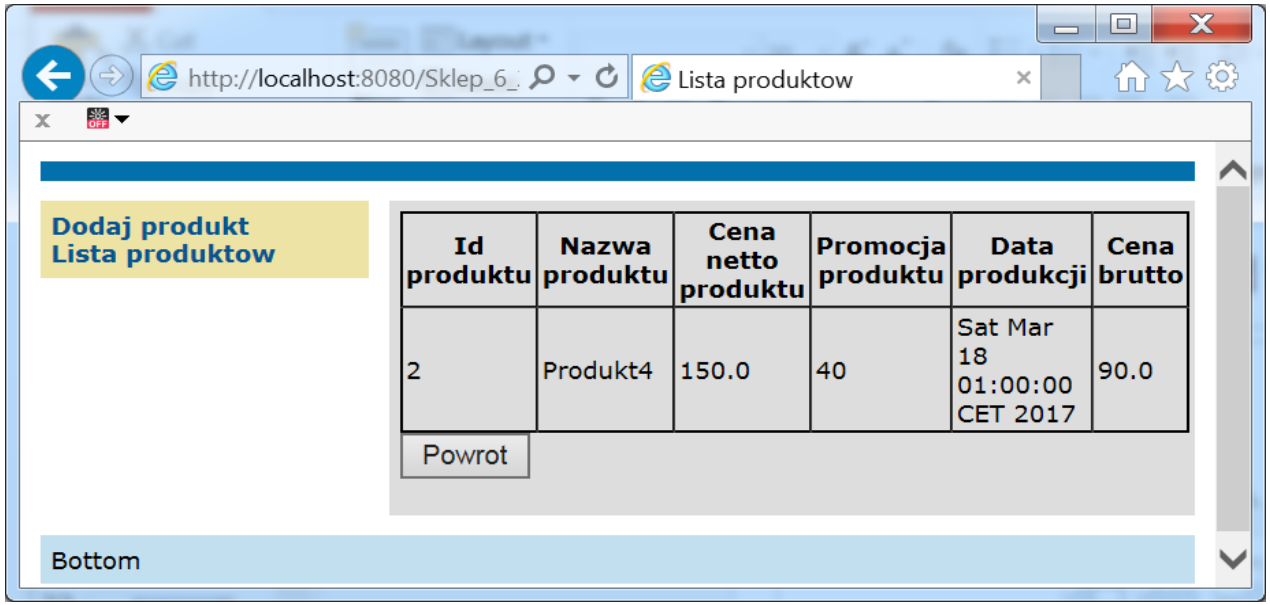


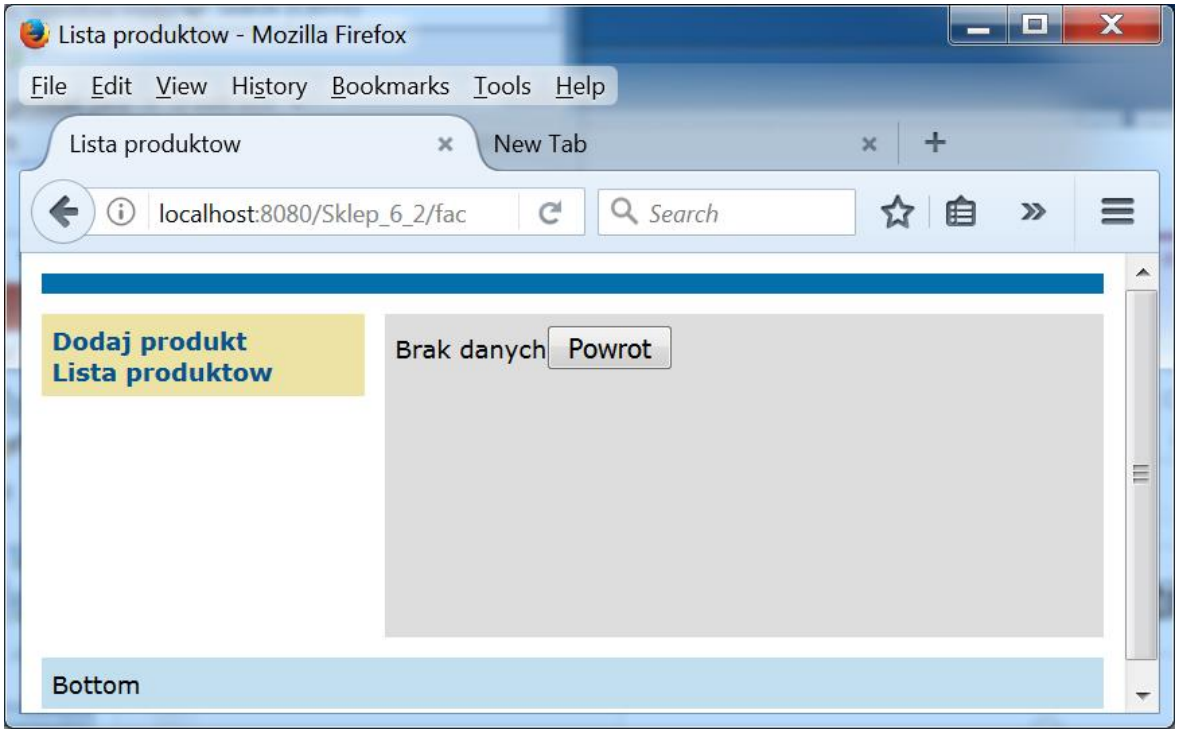


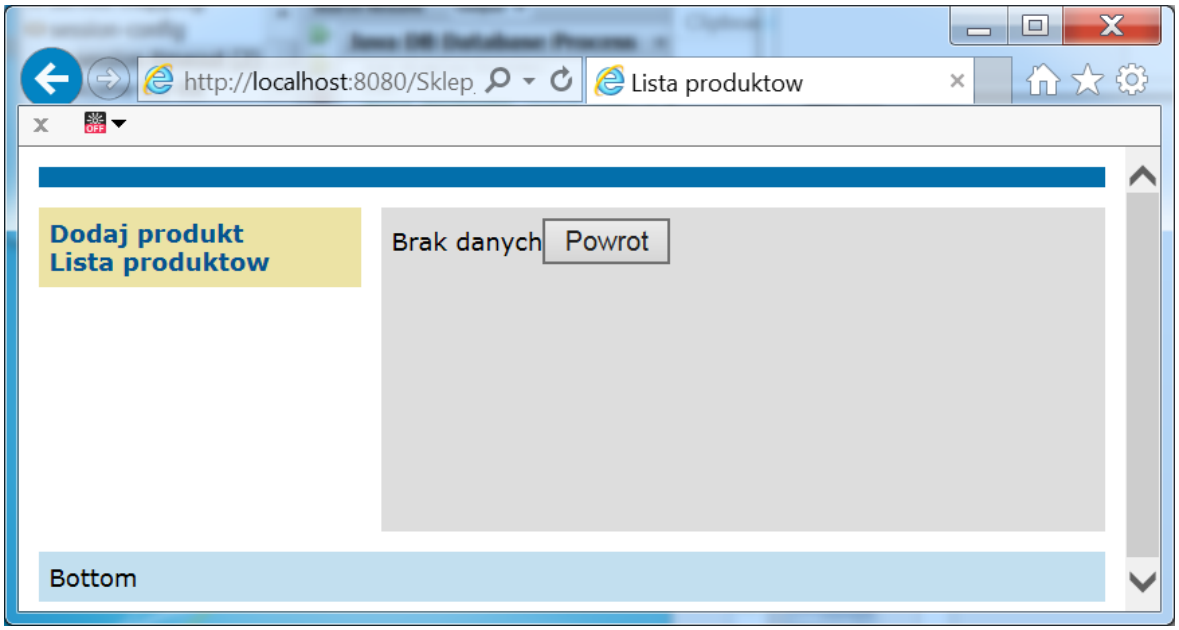


Po uruchomieniu formularza **Lista produktow** w obu instancjach klientów aplikacji widać, że korzystają **z różnych instancji** komponentu typu **Fasada_warstwy_biznesowej_ejb**, czyli różnych instancji obiektu **Fasada_warstwy_biznesowej.**





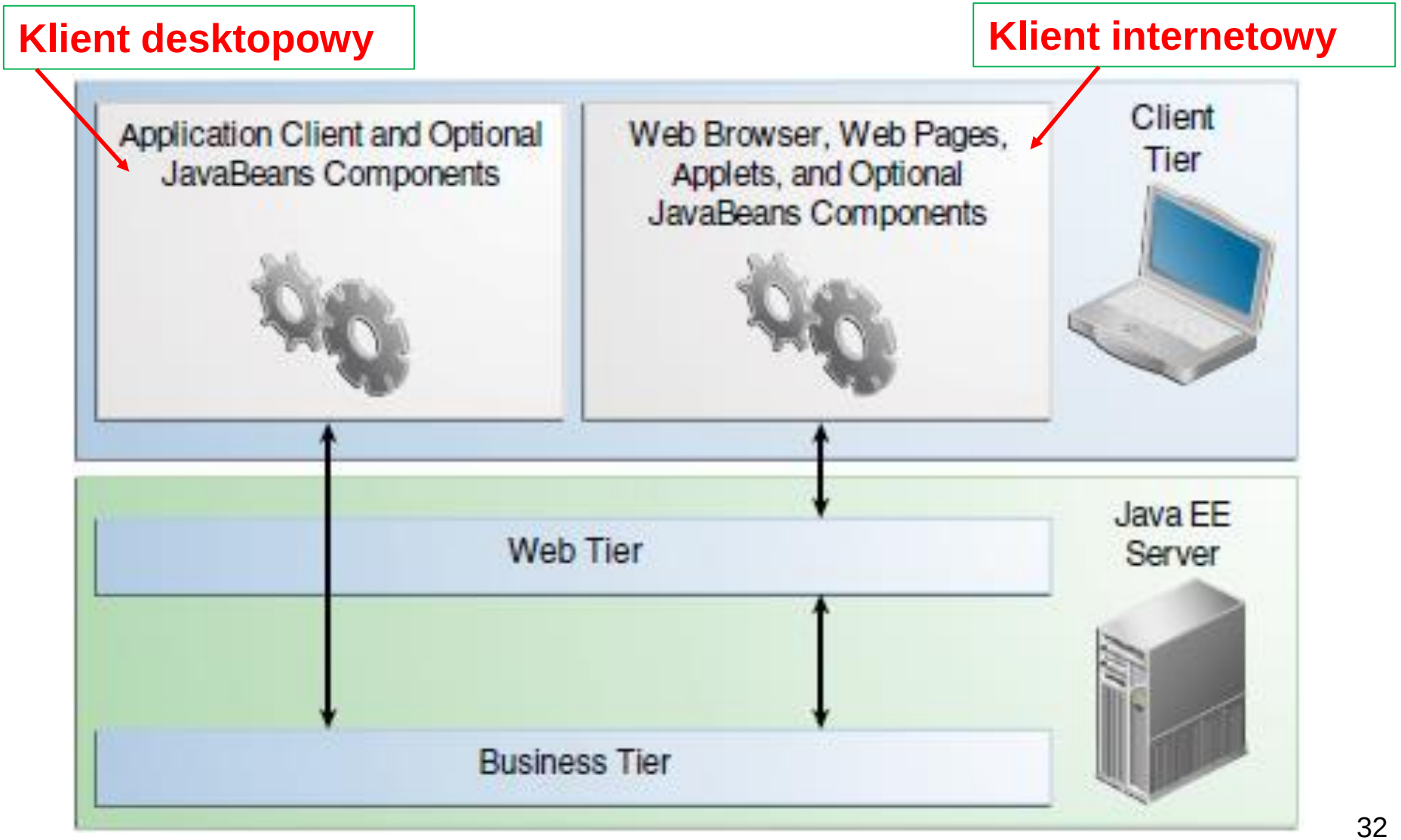




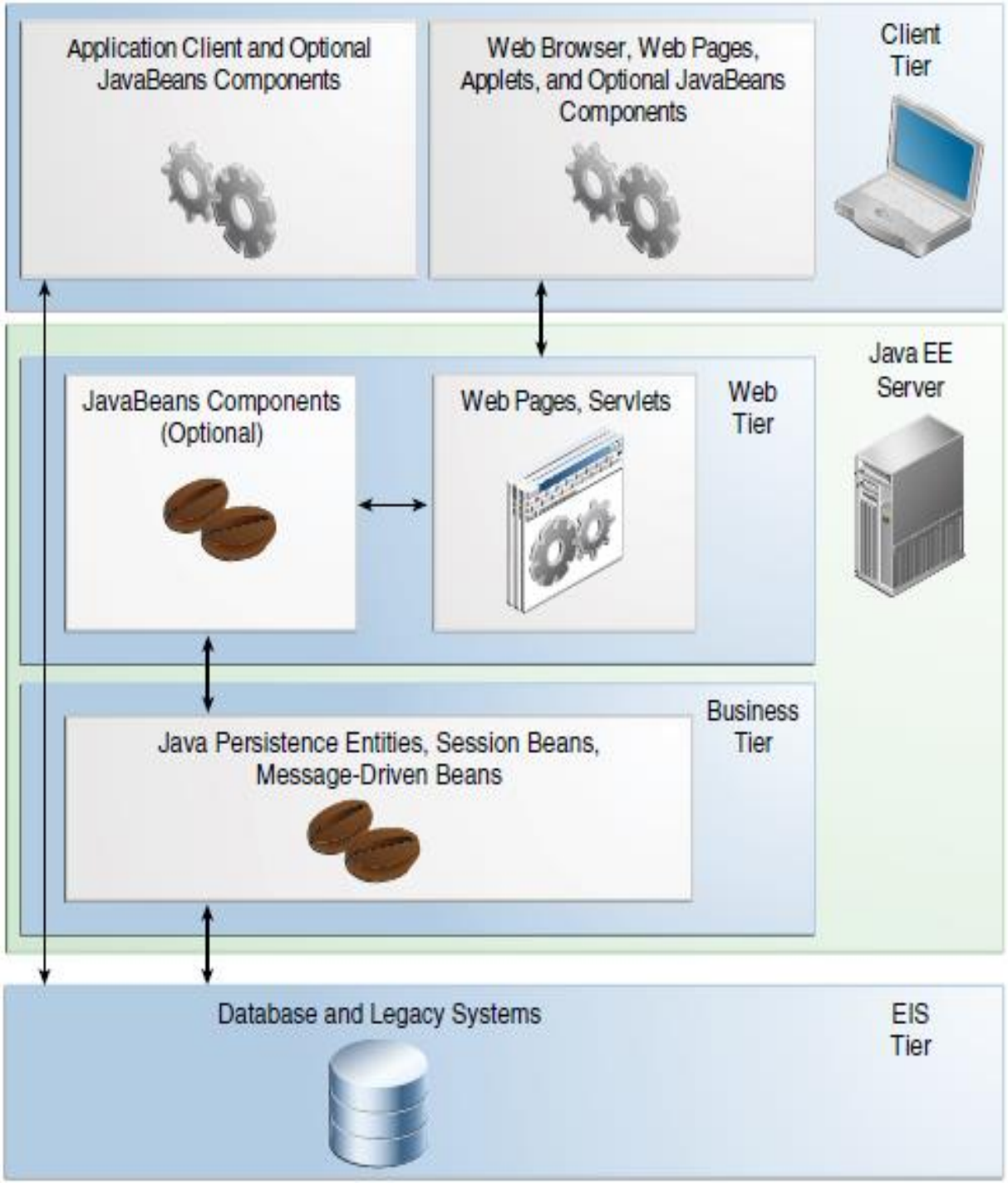
Po upływie 2 min próba wywołania strony Lista produktów za pomocą linku Lista produktów daje rezulat – Brak danych, ponieważ każdy z komponentów typu **Stateful** przeszedł w stan nieaktywny po upływie 2 min i nastąpiła utrata danych.

3. Wykonanie aplikacji wielowarstwowej na platformie Java EE z warstwą klienta zawierającą klienta internetowego i desktopowego

Komunikacja między warstwą klienta i serwerem aplikacji

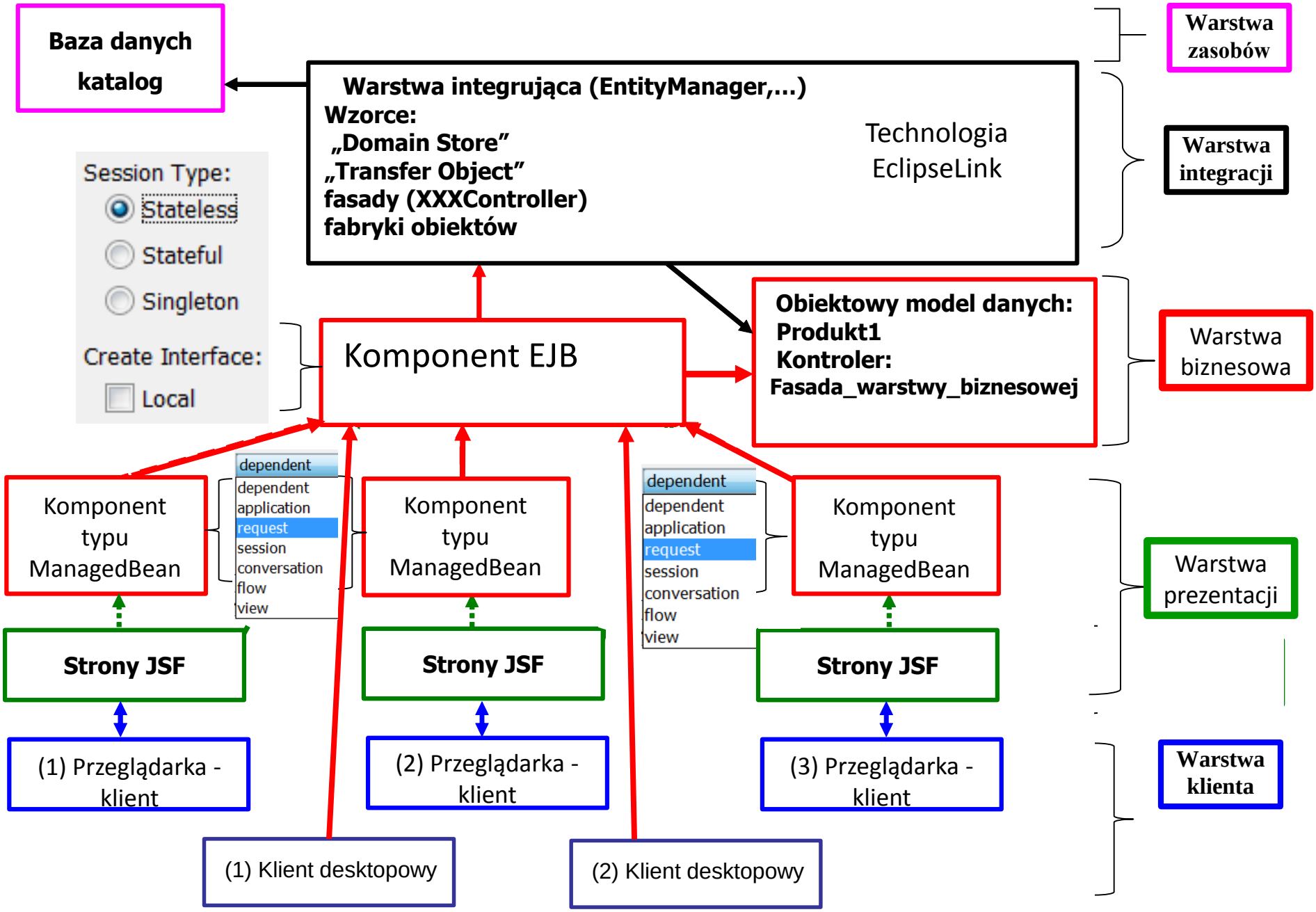


Komponenty poszczególnych warstw aplikacji EE

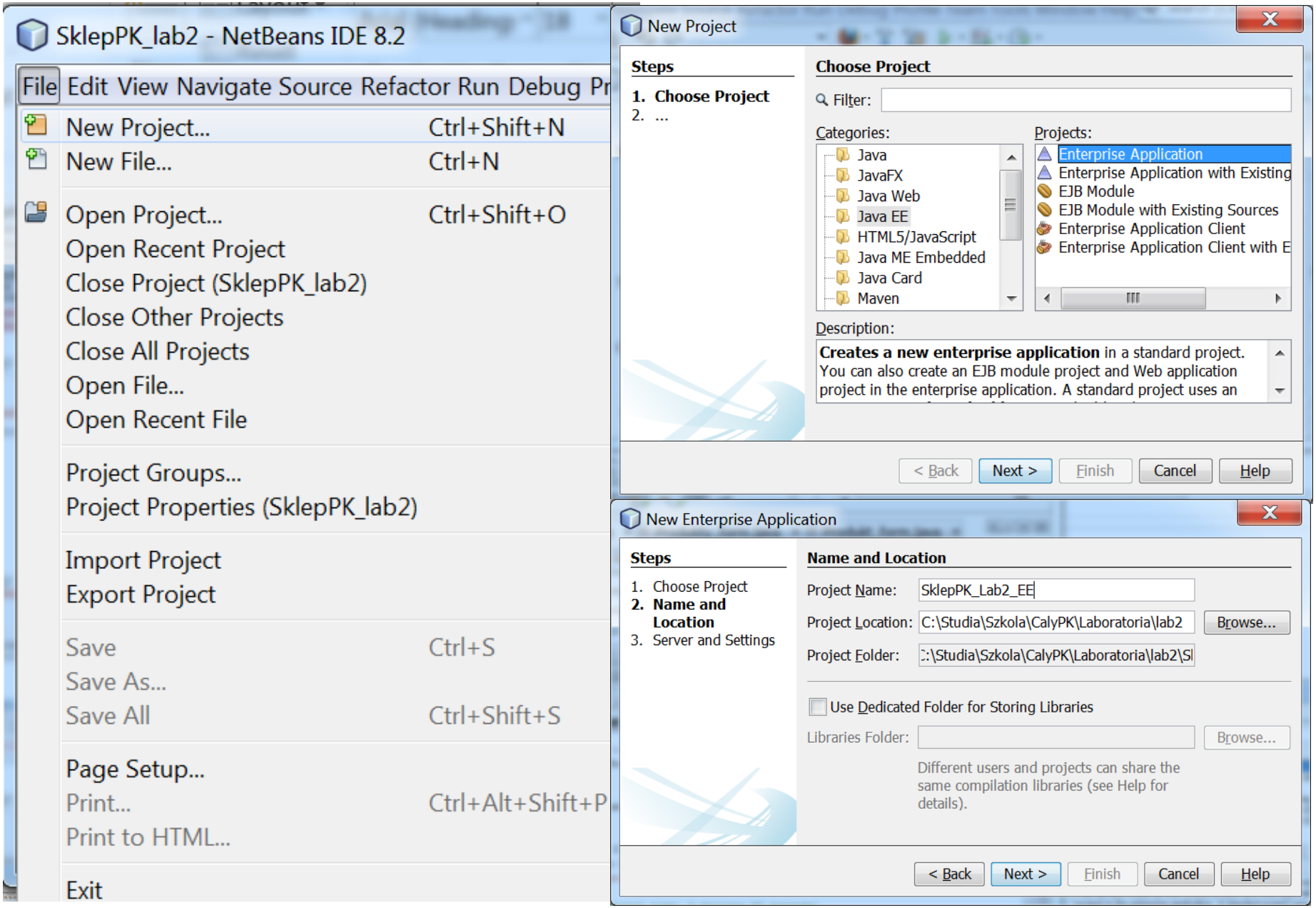


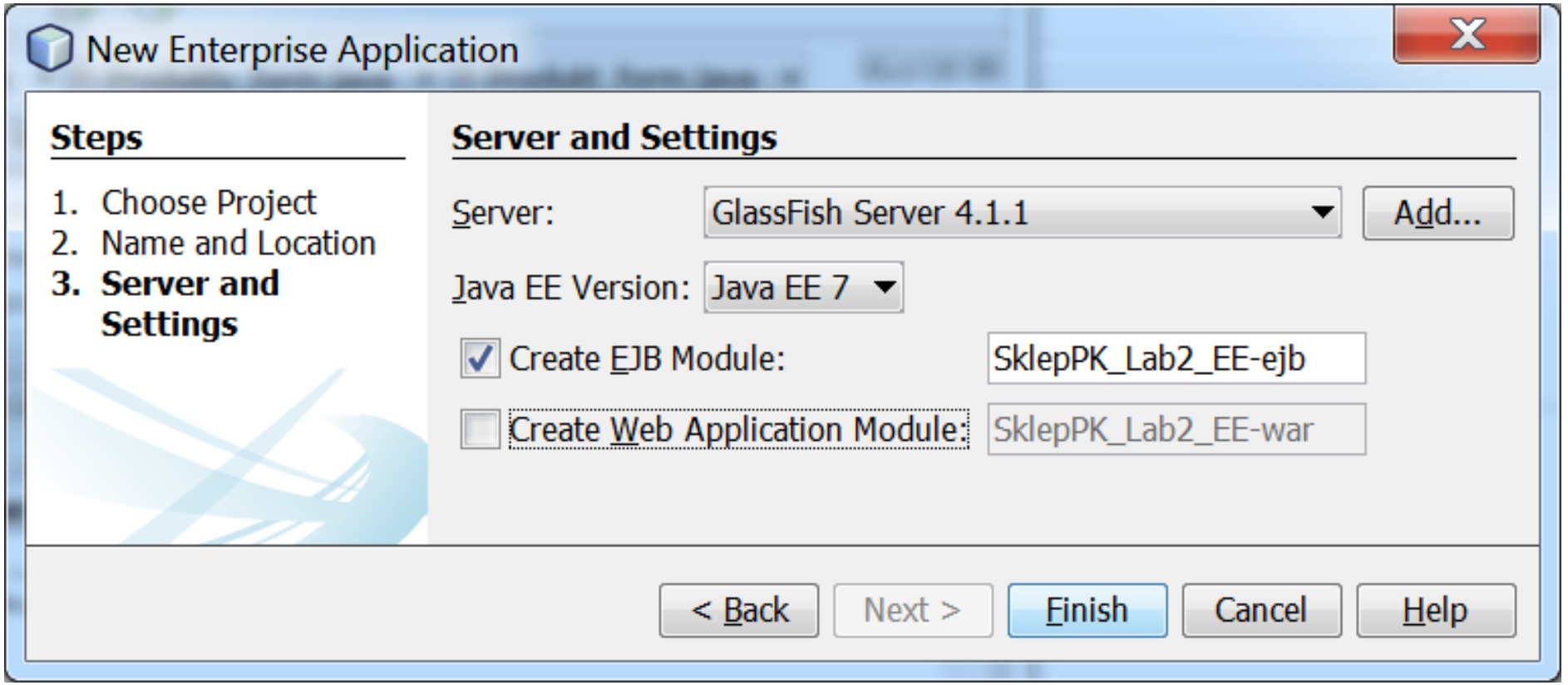
,

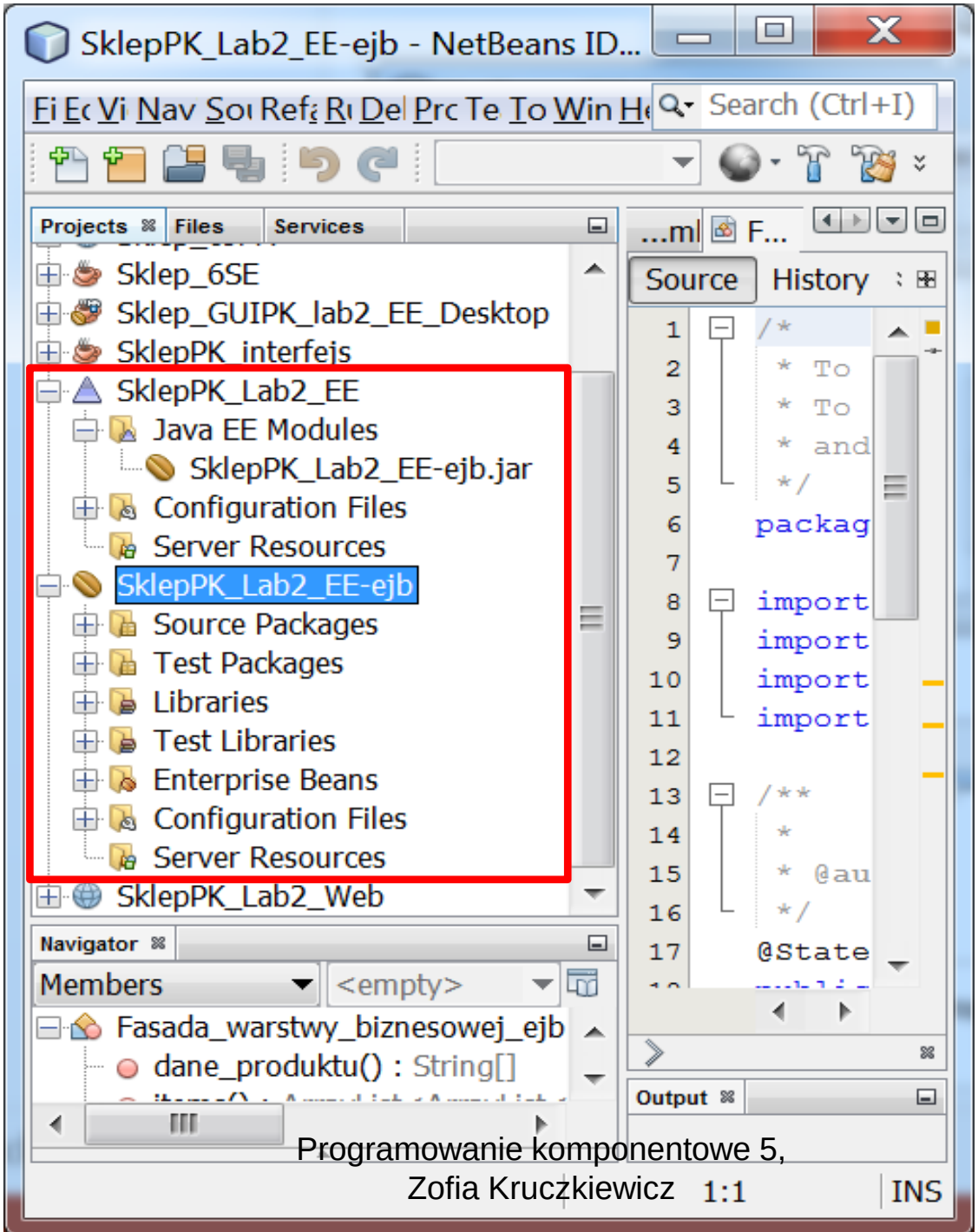
Architektura aplikacji pięciowarstwowej -Java EE 7.0 JavaServer Faces



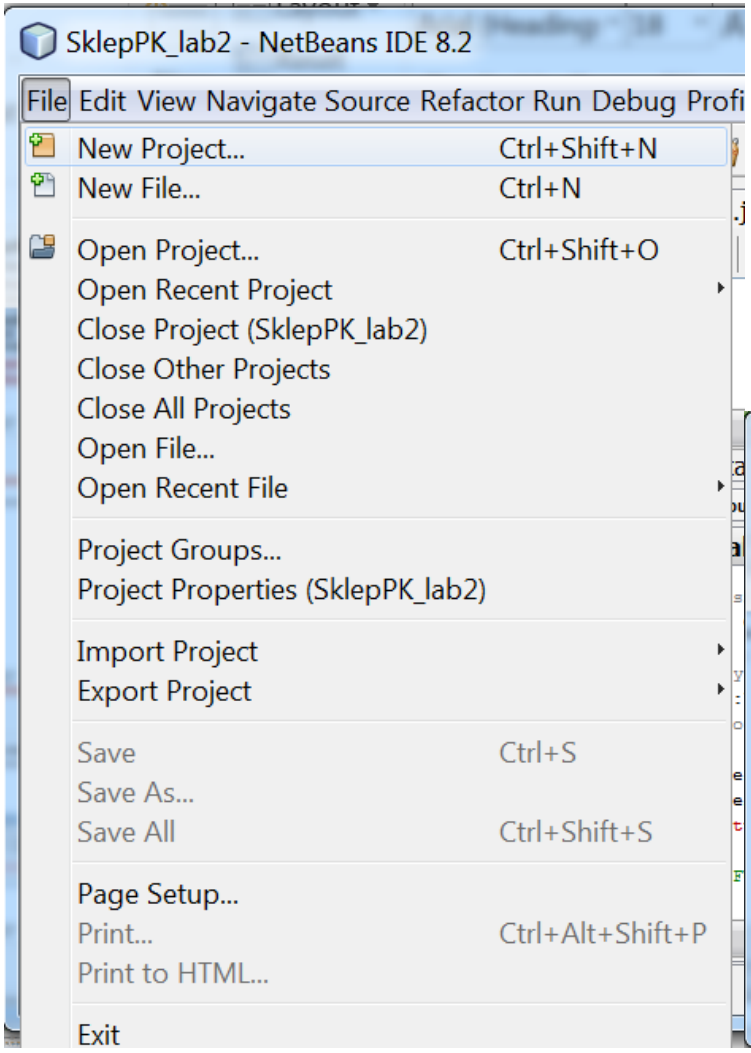
3.1. Wykonanie projektu typu Enterprise Application

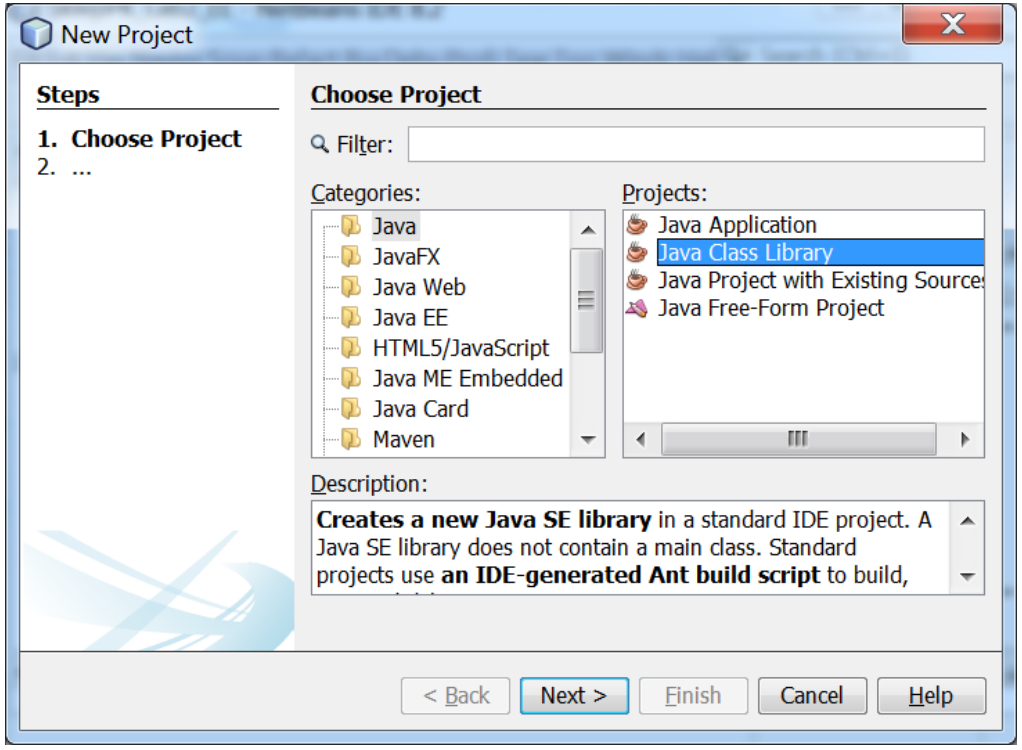


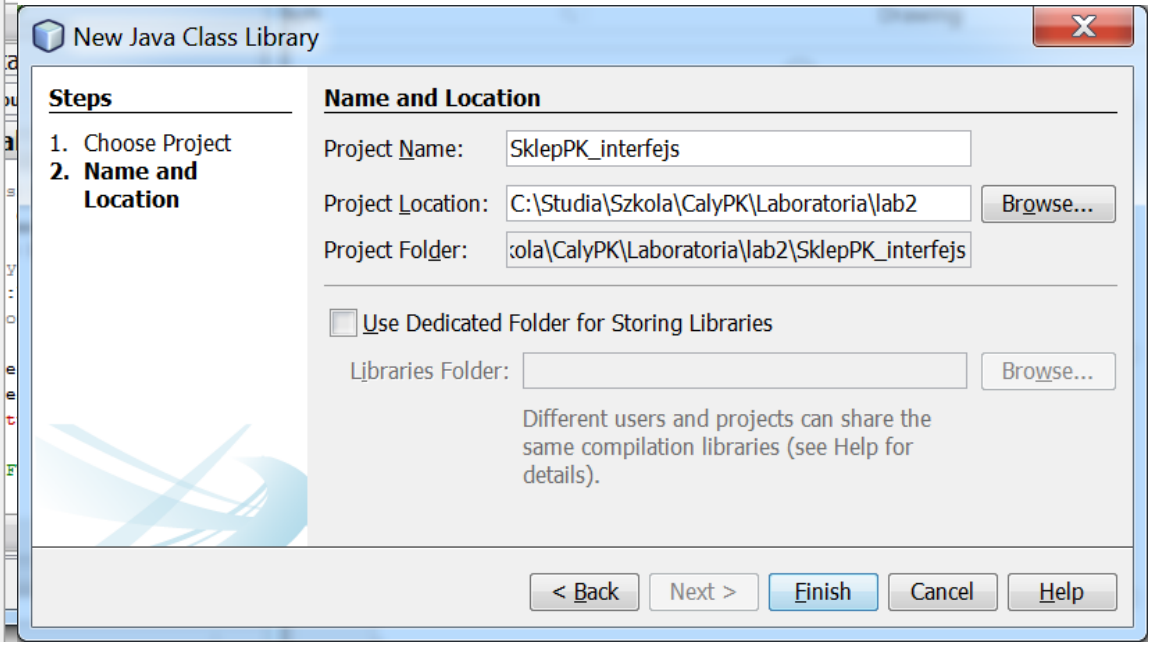

New Enterprise Application
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
Server and Settings
Server:
GlassFish Server 4.1.1
Add...
Java EE Version:
Java EE 7
Create EJB Module:
SklepPK_Lab2_EE-ejb
Create Web Application Module:
SklepPK_Lab2_EE-war
< Back
Next >
Finish
Cancel
Help


SklepPK_Lab2_EE-ejb - NetBeans ID...
Search (Ctrl+I)
Projects
Files
Services
Sklep_6SE
Sklep_GUIPK_lab2_EE_Desktop
SklepPK_interfejs
SklepPK_Lab2_EE
Java EE Modules
SklepPK_Lab2_EE-ejb.jar
Configuration Files
Server Resources
SklepPK_Lab2_EE-ejb
Source Packages
Test Packages
Libraries
Test Libraries
Enterprise Beans
Configuration Files
Server Resources
SklepPK_Lab2_Web
Navigator
Members
<empty>
Fasada_warstwy_biznesowej_ejb
dane_produktu() : String[]
Source
History
/*
* To
* To
* and
*/
packag
import
import
import
import
/**
*
* @au
*/
@State
Output
1:1
INS

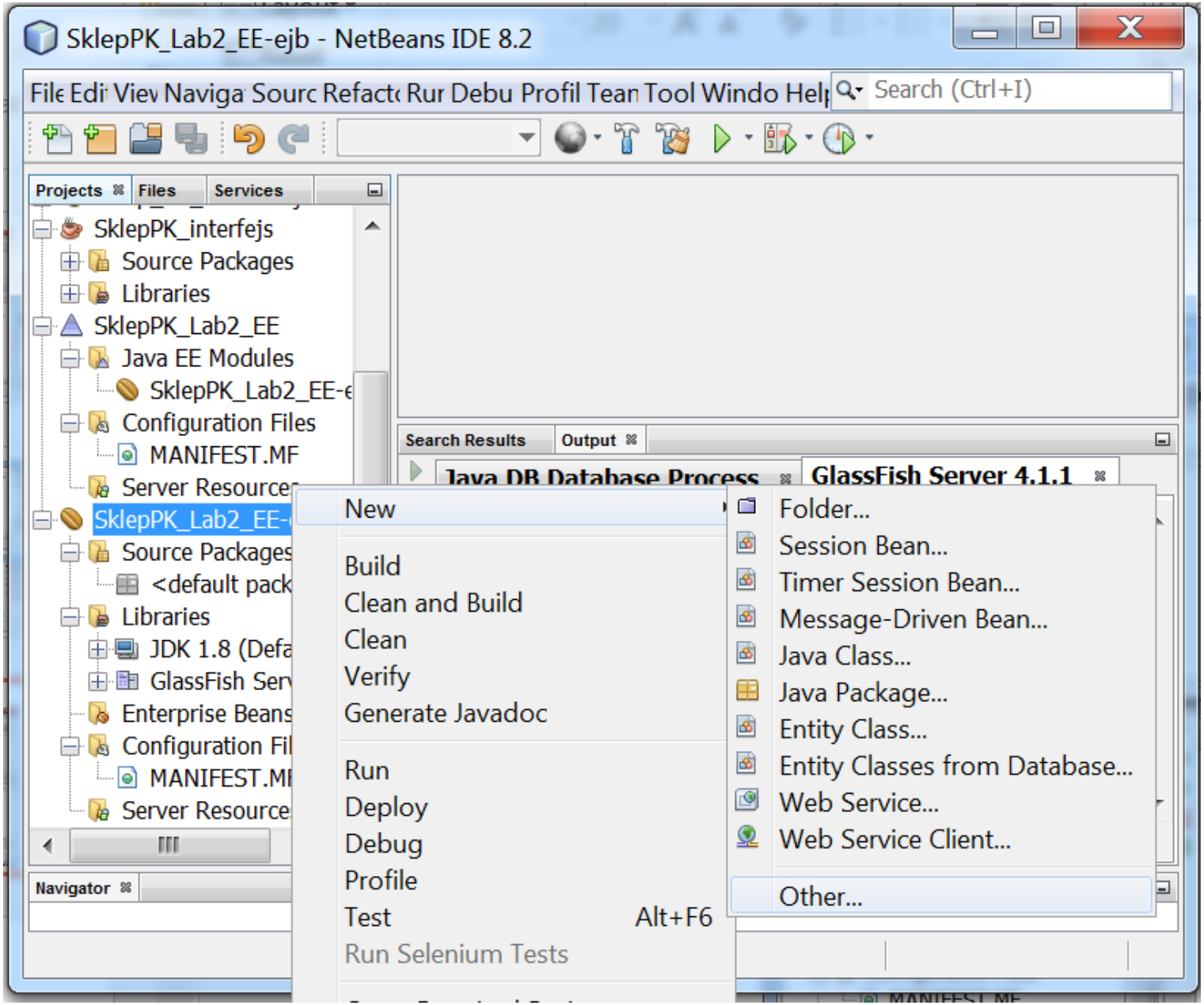
3.2. Wykonanie projektu Java SE typu Java Class Library (**SklepPK_interfejs**) w celu zdefiniowania w nim w kolejnych krokach **interfejsu komponentu EJB**.

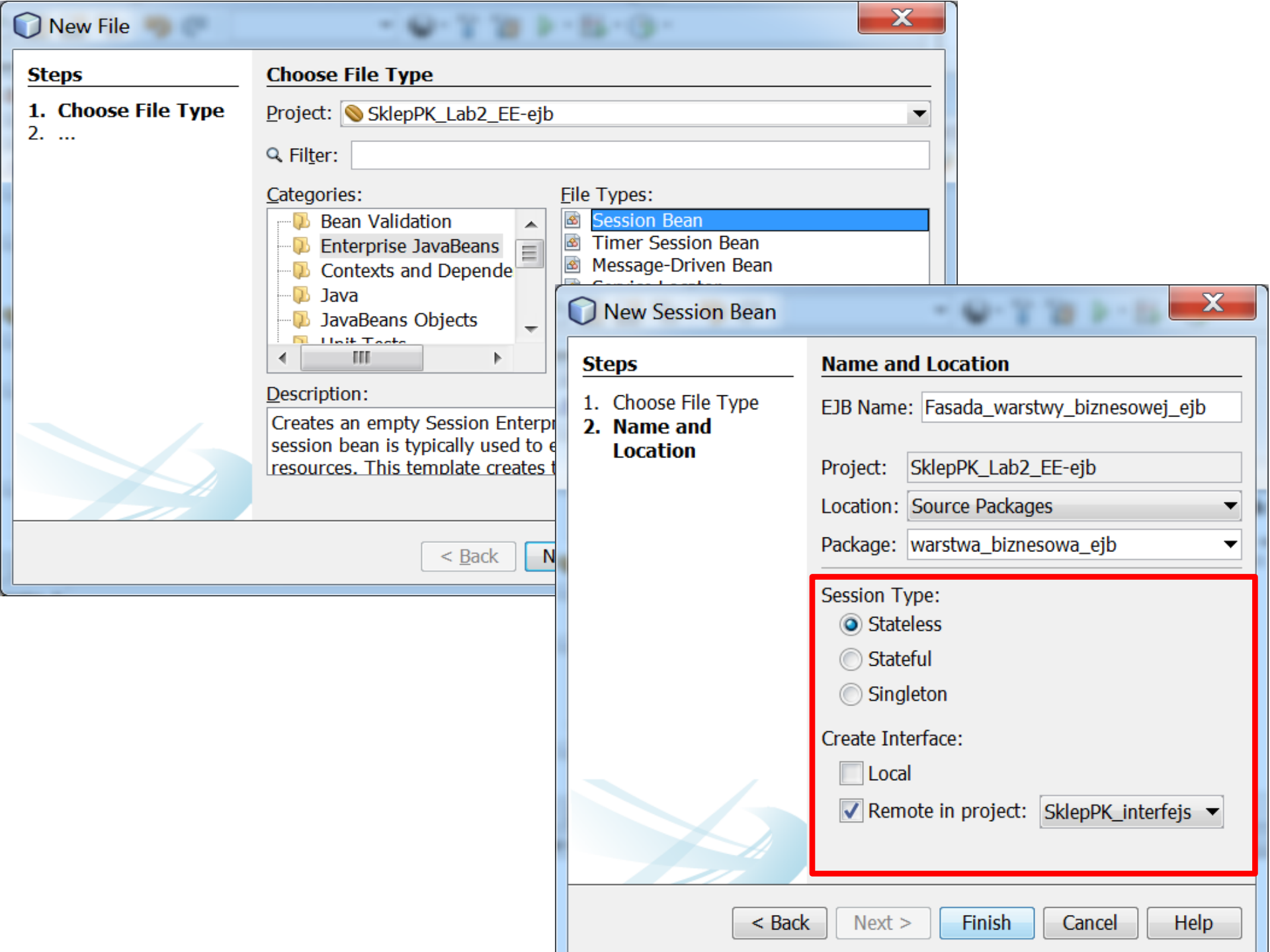




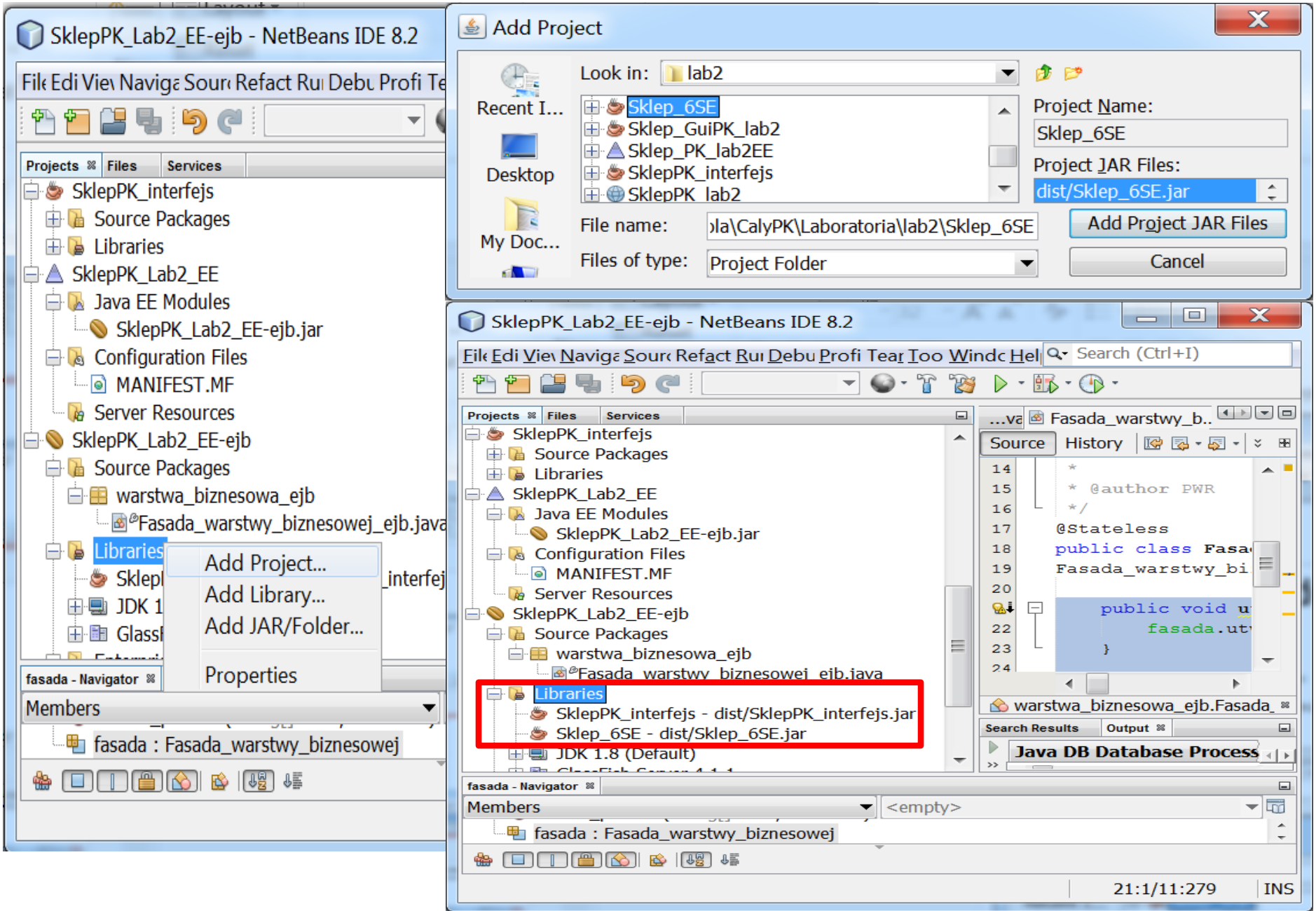


3.3. Definicja komponentu EJB typu Session w module typu EJB (wybranie pozycji **New/Other**)

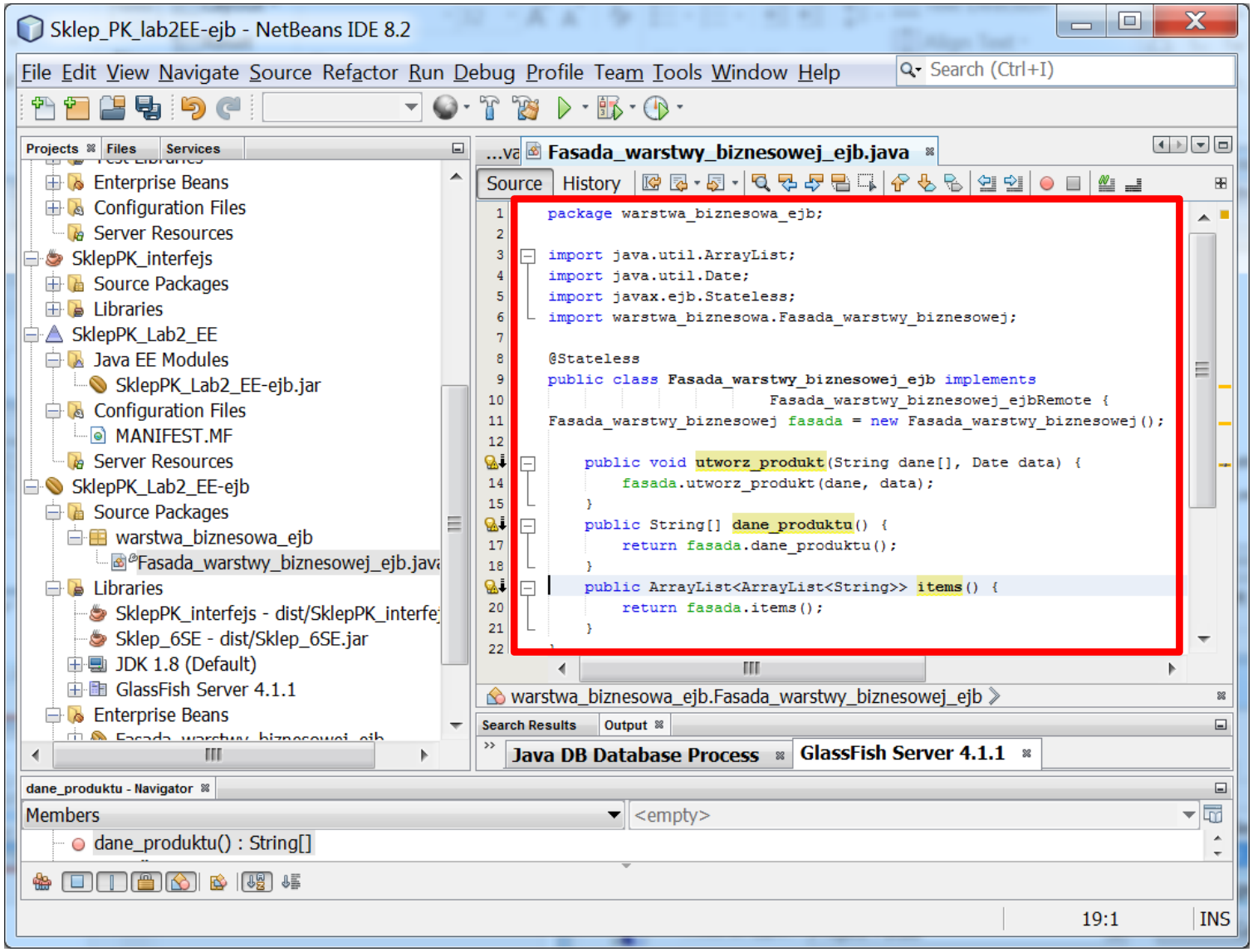



New File
Steps
1. Choose File Type
2. ...
Choose File Type
Project: SklepPK_Lab2_EE-ejb
Filter:
Categories:
Bean Validation
Enterprise JavaBeans
Contexts and Depende
Java
JavaBeans Objects
File Types:
Session Bean
Timer Session Bean
Message-Driven Bean
Description:
Creates an empty Session Enterpr
session bean is typically used to e
resources. This template creates t
< Back
New Session Bean
Steps
1. Choose File Type
2. Name and Location
Name and Location
EJB Name: Fasada_warstwy_biznesowej_ejb
Project: SklepPK_Lab2_EE-ejb
Location: Source Packages
Package: warstwa_biznesowa_ejb
Session Type:
Stateless
Stateful
Singleton
Create Interface:
Local
Remote in project: SklepPK_interfejs
< Back
Next >
Finish
Cancel
Help

3.4. Dodanie projektu SE z logiką biznesową do modułu EJB



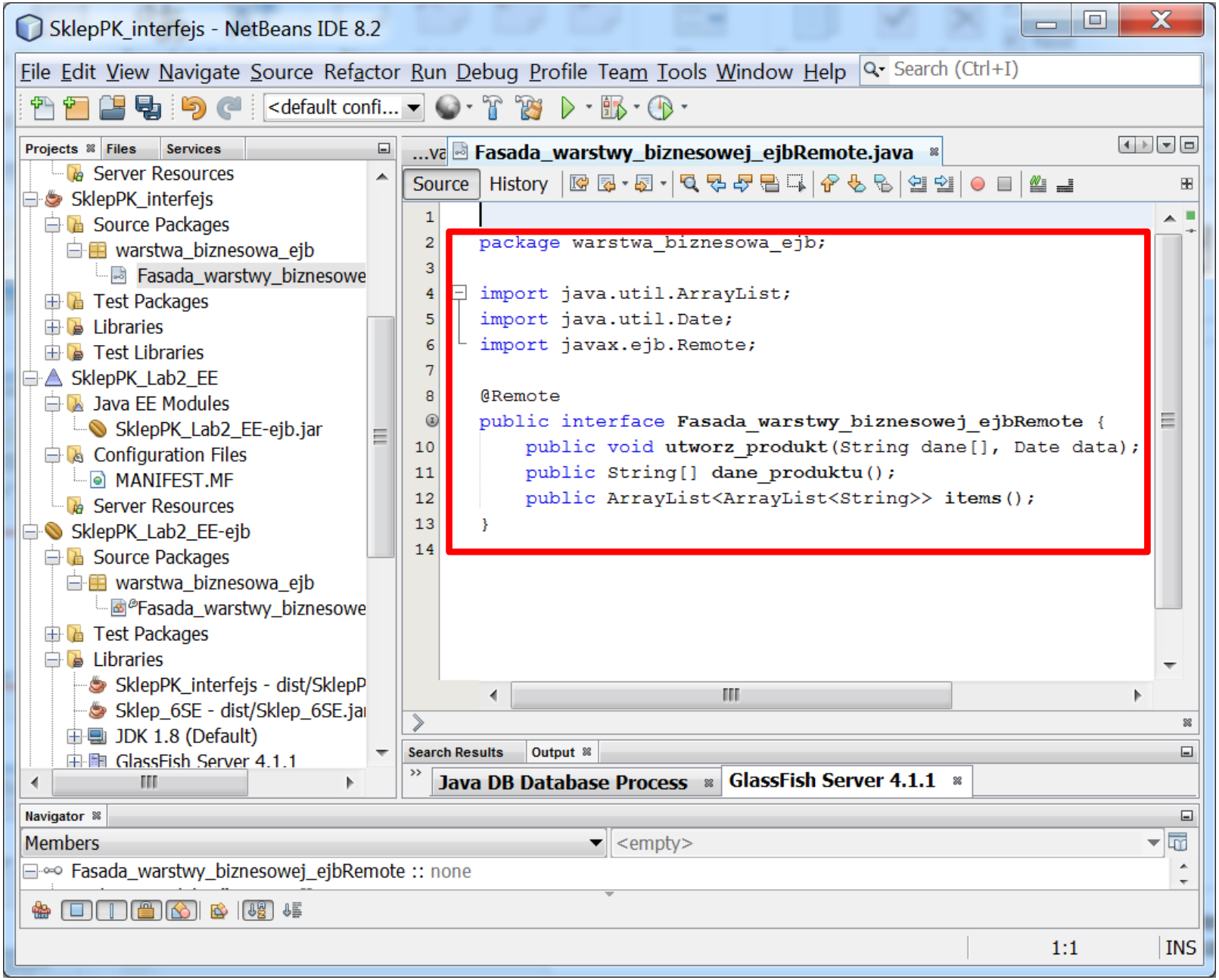
3.5. Zdefiniowanie w komponencie EJB typu Session Bean (Fasada_warstwy_biznesowej_ejb) kodu zdalnego dostępu do logiki biznesowej



Fasada_warstwy_biznesowej_ejb - kod

```
package warstwa_biznesowa;

import java.util.ArrayList;
import java.util.Date;
import javax.ejb.Stateless;
import warstwa_biznesowa.Fasada_warstwy_biznesowej;

@Stateless
public class Fasada_warstwy_biznesowej_ejb implements
                                        Fasada_warstwy_biznesowej_ejbRemote {
 Fasada_warstwy_biznesowej fasada=new Fasada_warstwy_biznesowej();

    public void utworz_produkt(String dane[], Date data) {
        fasada.utworz_produkt(dane, data);
       }

    public String[] dane_produktu() {
         return fasada.dane_produktu(); }

    public ArrayList<ArrayList<String>> items() {
         return fasada.items(); }
}
```

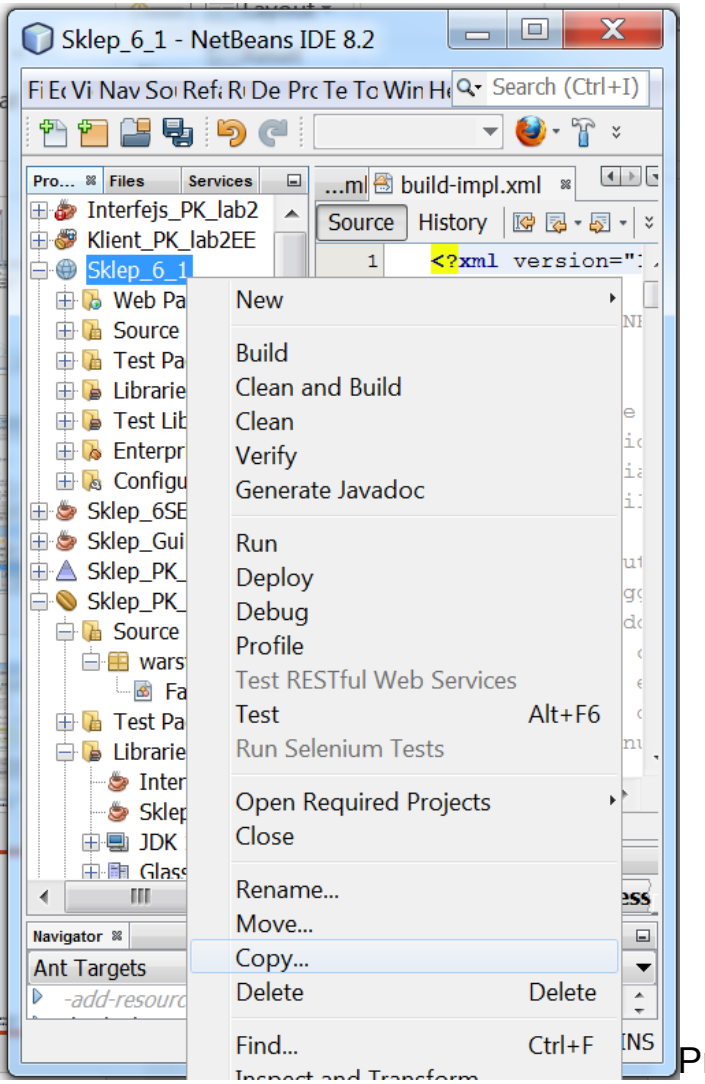
Kod interfejsu – dla zdalnego komponentu typu EJB

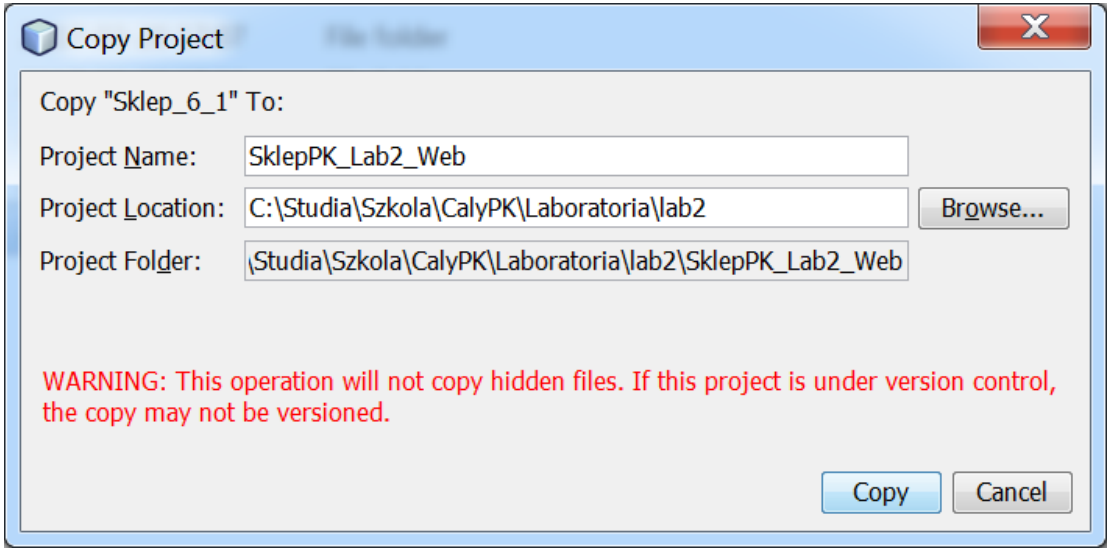


Fasada_warstwy_biznesowej_ejbRemote – kod interfejsu

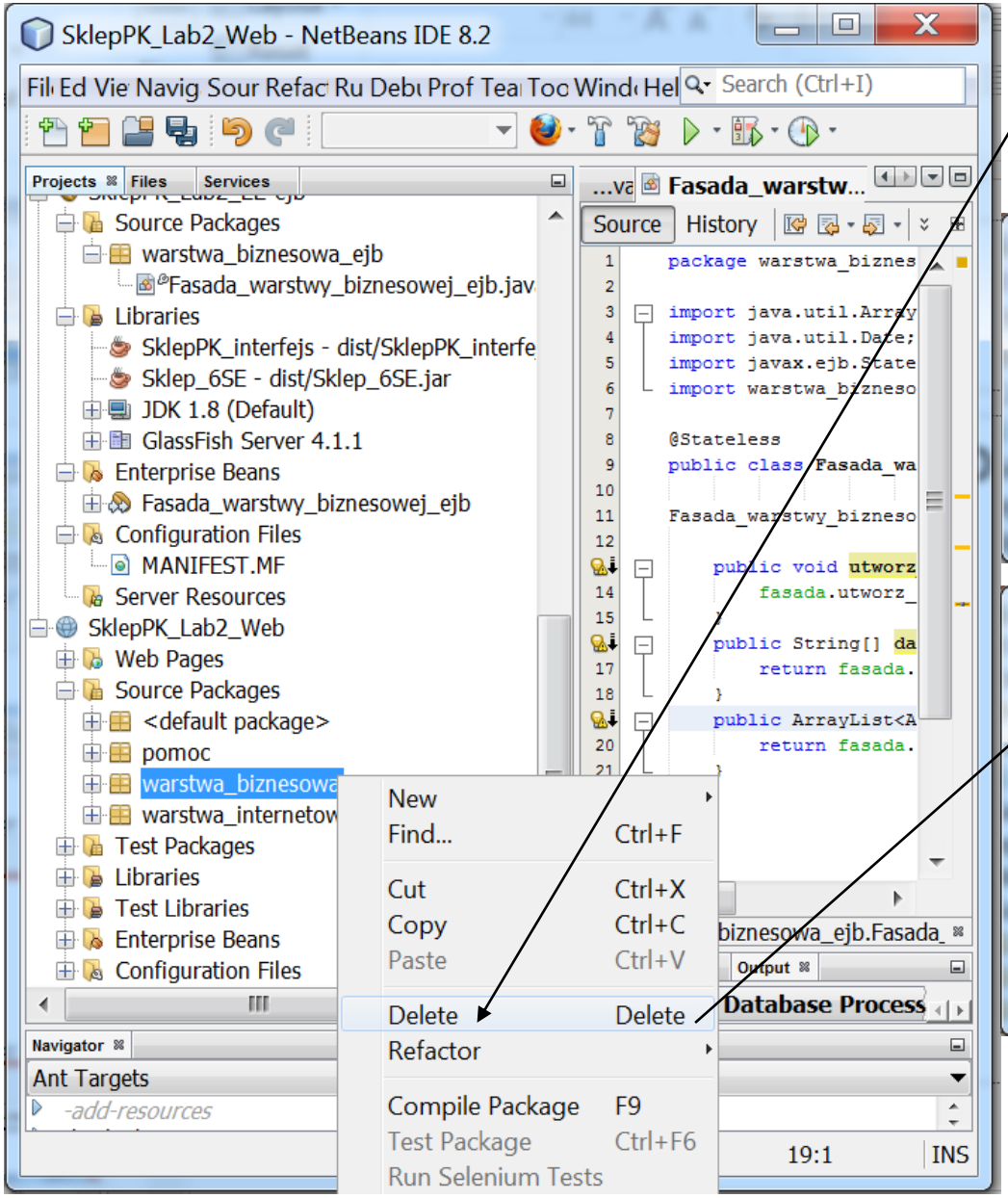
```
package warstwa_biznesowa_ejb;

import java.util.ArrayList;
import java.util.Date;
import javax.ejb.Remote;

@Remote
public interface Fasada_warstwy_biznesowej_ejbRemote {
    public void utworz_produkt(String dane[], Date data);
    public String[] dane_produktu();
    public ArrayList<ArrayList<String>> items();
}
```

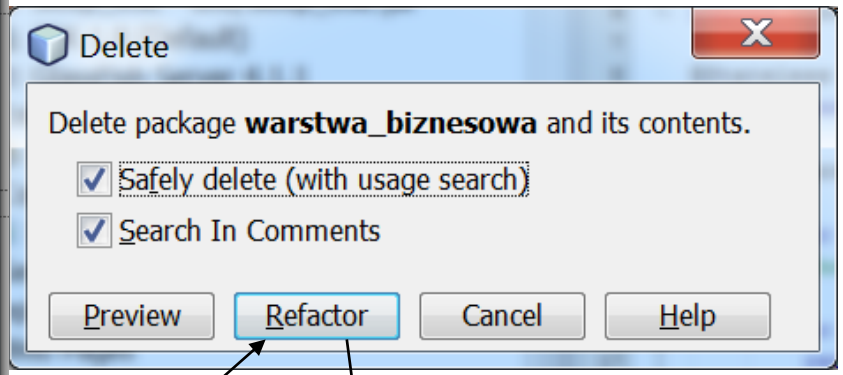
3.6. Dodanie do projektu EE warstwy klienta internetowego – kopia istniejącej aplikacji Java Web Application

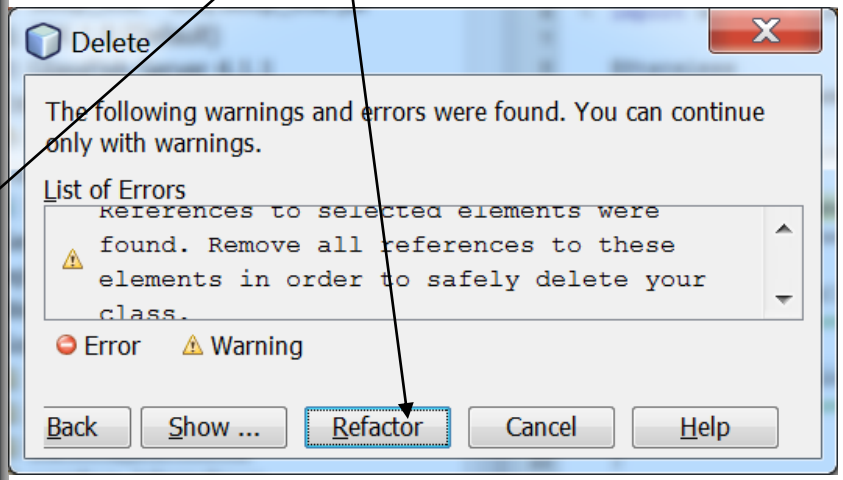




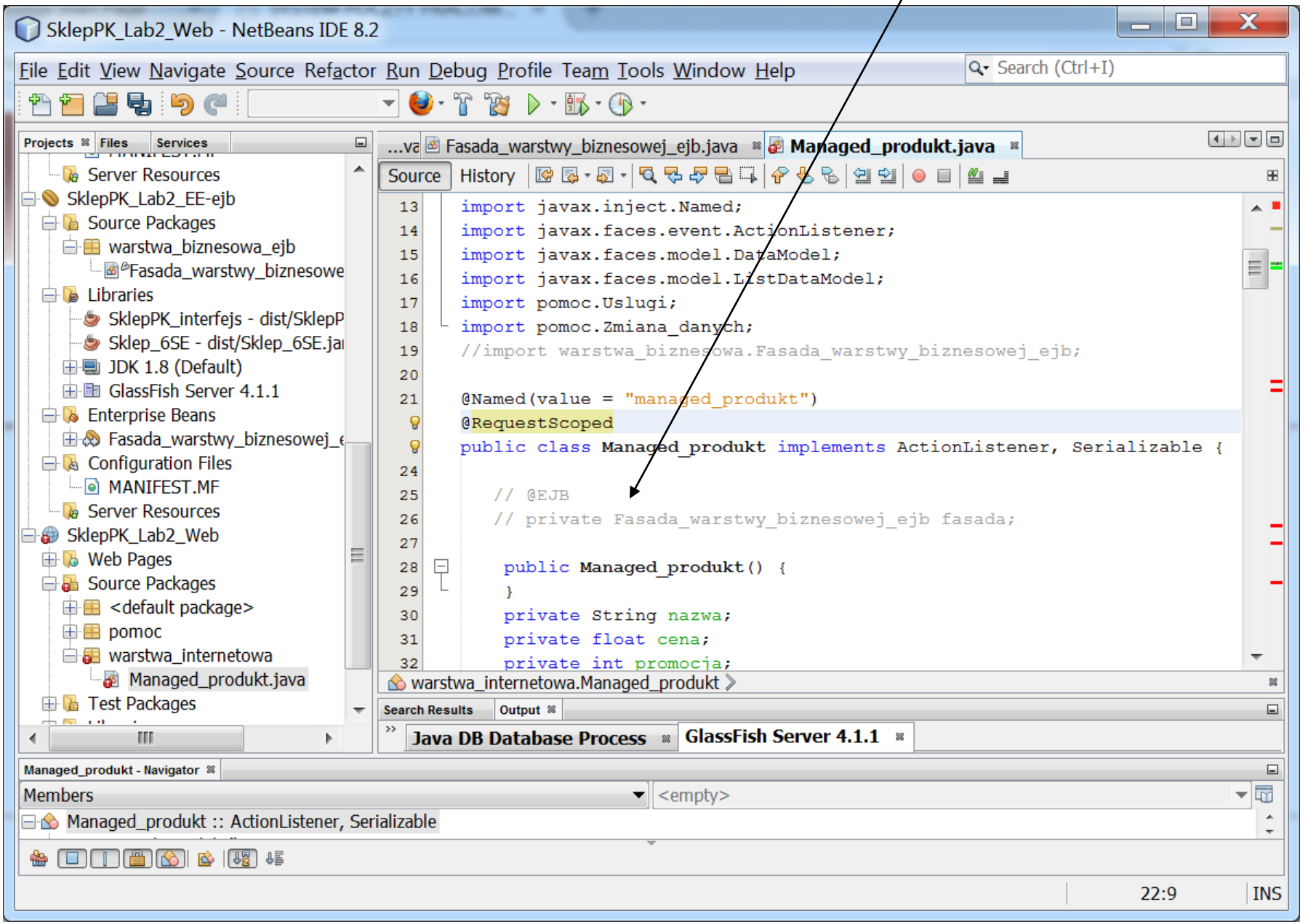
3.6.1. W projecie typu Java Web typu **Web Application** bezpiecznie pakiet **warstwa_biznesowa – teraz komponent EJB jest zdefiniowany w module EJB**



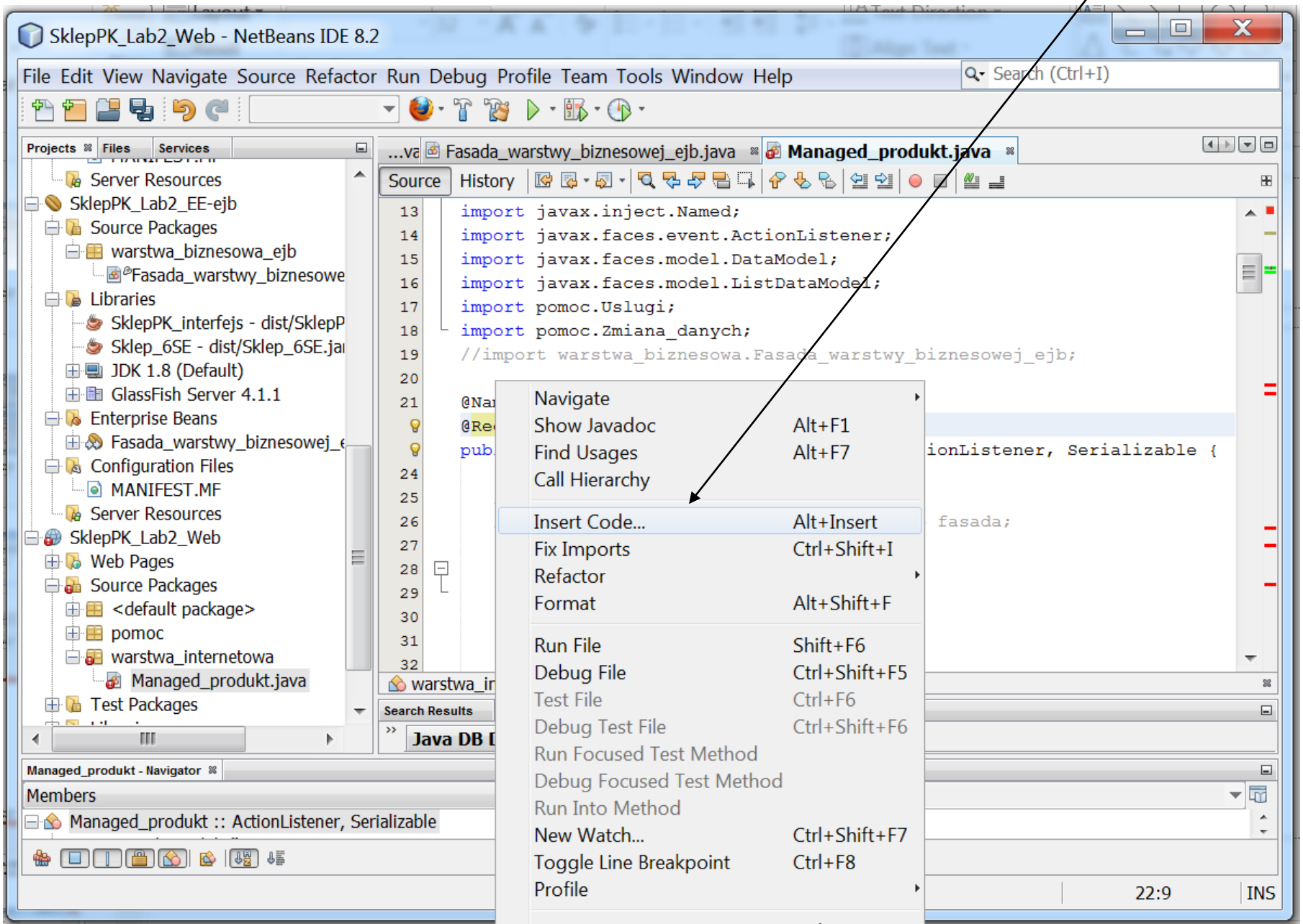


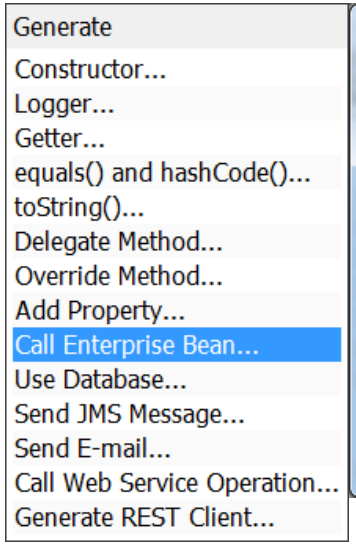


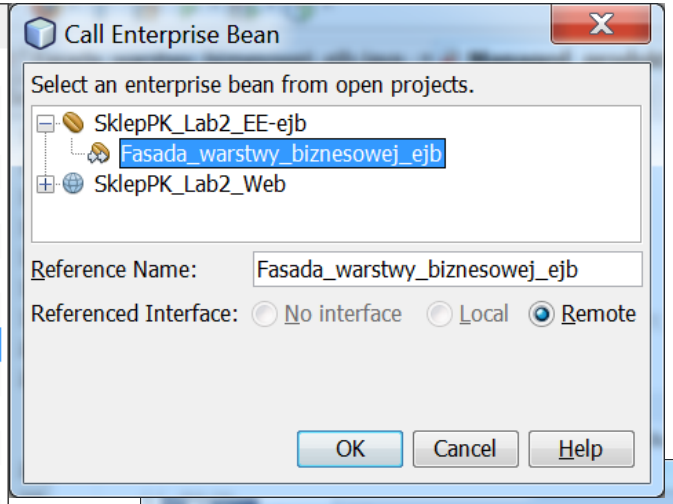
3.6.2. W komponencie typu Managed Bean o nazwie **Managed_produkt** należy usunąć z kodu dotychczasowe powiązanie z komponentem EJB.



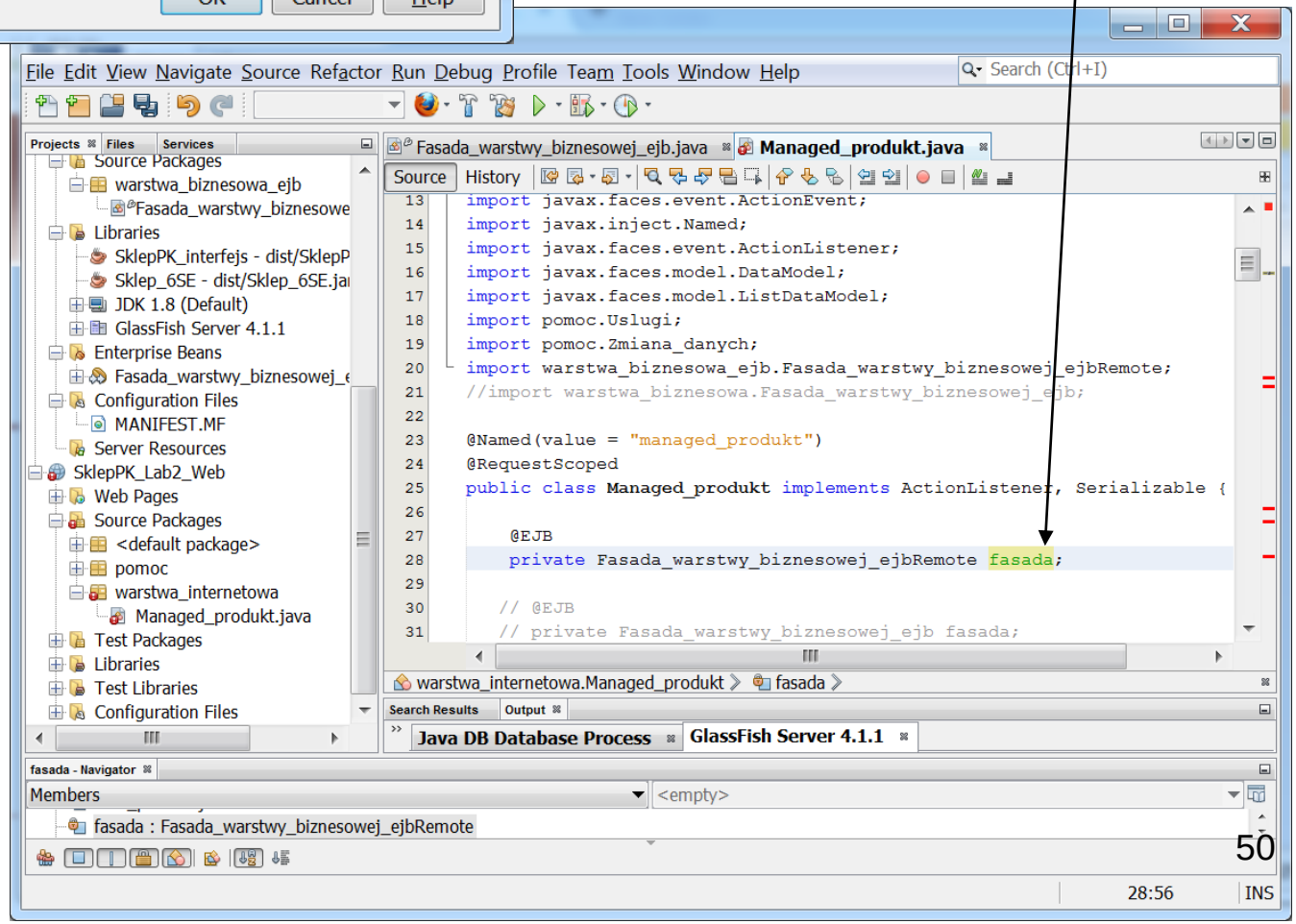
3.6.3. Należy teraz wstawić powiązanie z nowym komponentem typu EJB. W tym celu kliknąć prawym klawiszem myszy na kod klasy i wybrać pozycję **Insert Code...**



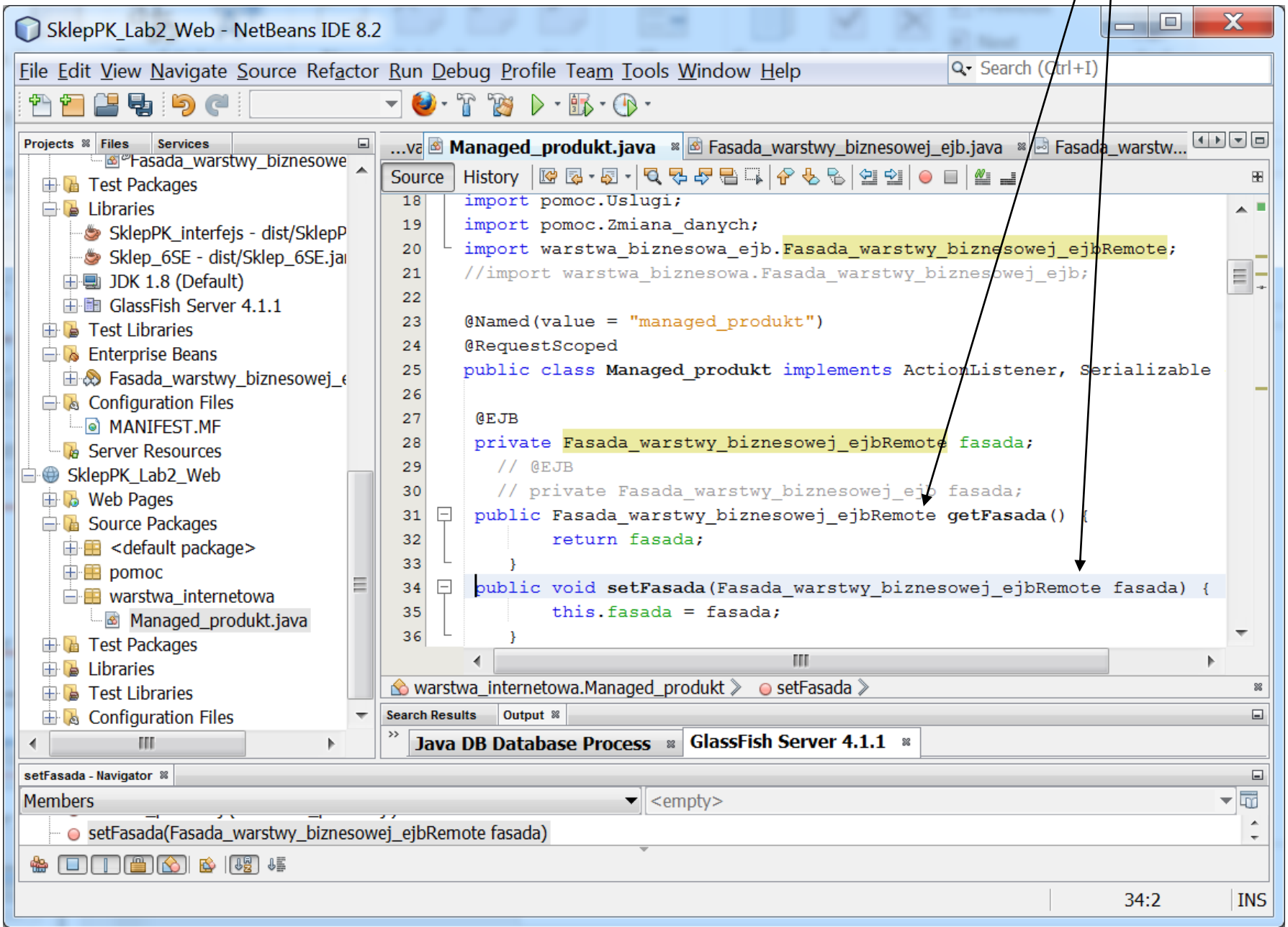




3.6.3.cd. Należy wybrać pozycję **Call Enterprise Bean..** i w kolejnym formularzu należy wybrać w projekcie EJB **SklepPK_Lab2_EE-ejb** komponent EJB typu Session: **Fasada_warstwy_biznesowej_ejbRemote i** nadać mu nazwę **fasada.**

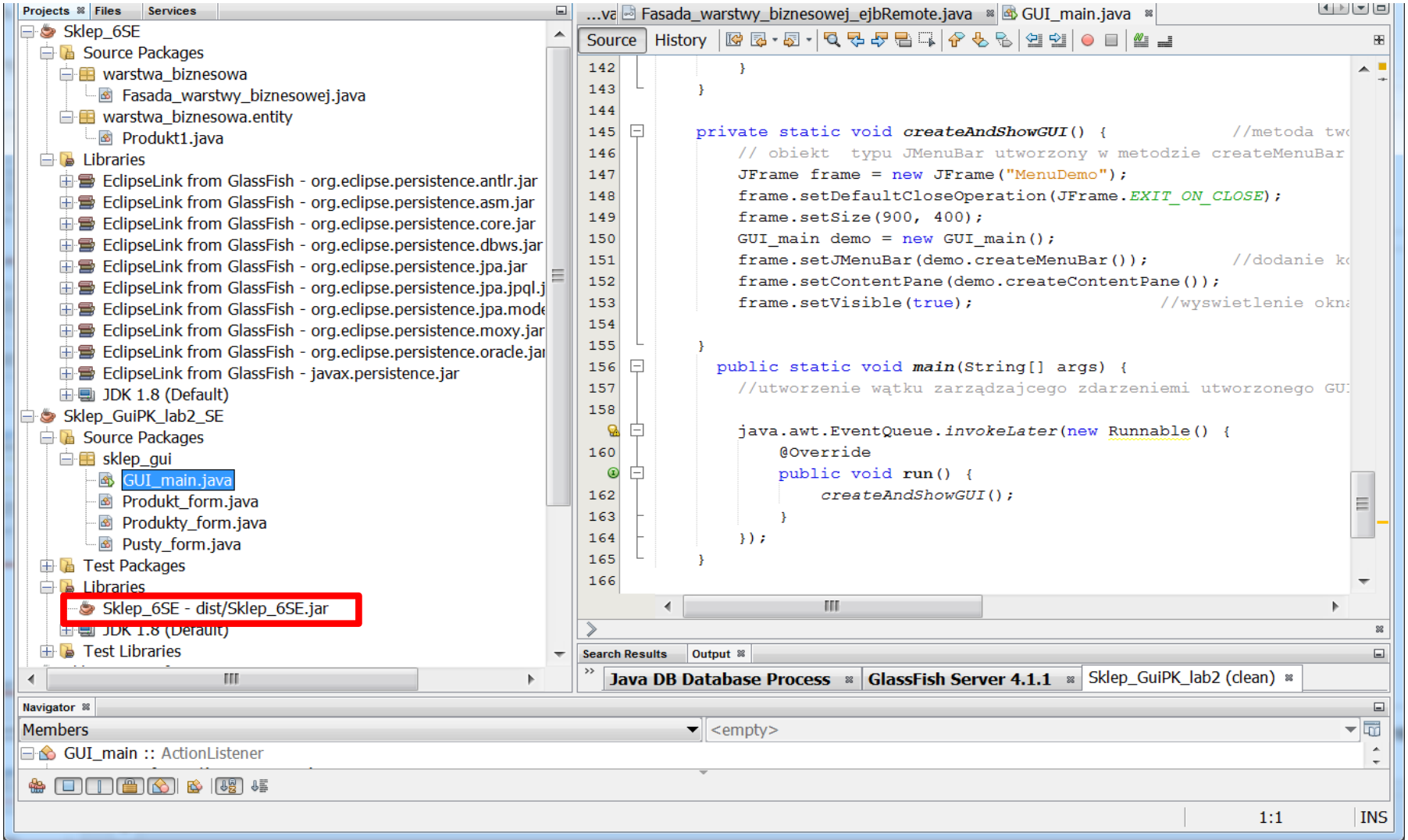


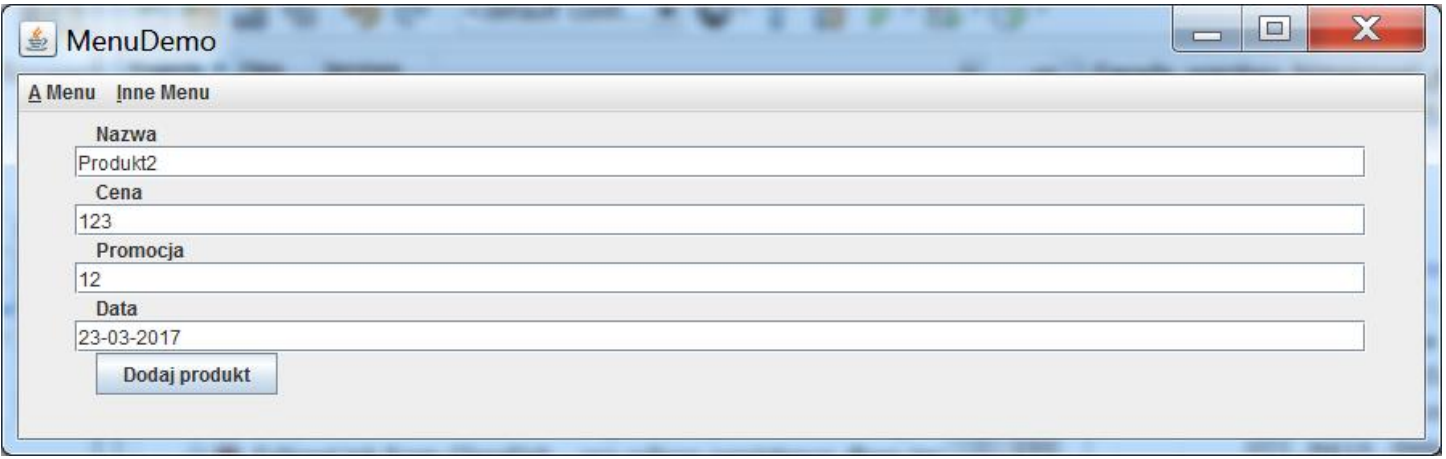
3.6.4. Wynik zmian – w celu usunięcia błędów należy zmodyfikować metody dostępu do dodanego komponentu:**Fasada_warstwy_biznesowej_ejbRemote**

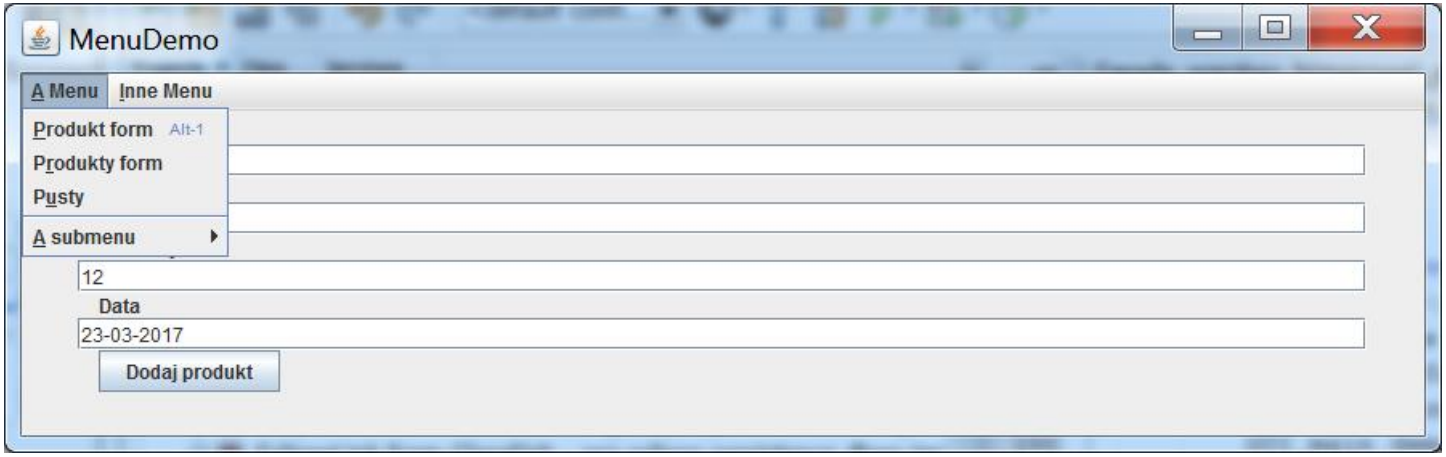


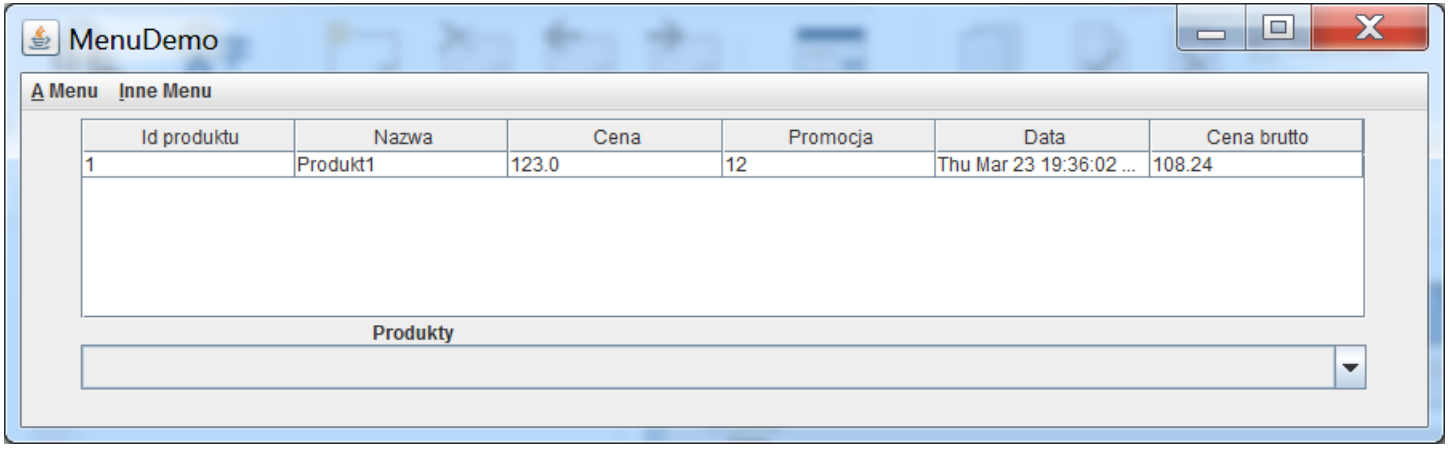
3.7. Dodanie warstwy klienta desktopowego Java EE typu Enterprise Application Client. Pierwszym etapem jest utworzenie projektu na platformie Java SE (http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Lab6_2016.pdf).

Logika biznesowa dla projektu **Sklep_GuiPK_lab2_SE** stanowiącego warstwę klienta w dwuwarstwowej aplikacji Java SE jest projekt **Sklep_6SE – z kodem logiki biznesowej (p.3.4 - połączonej podobnie jak z modułem EJB)**





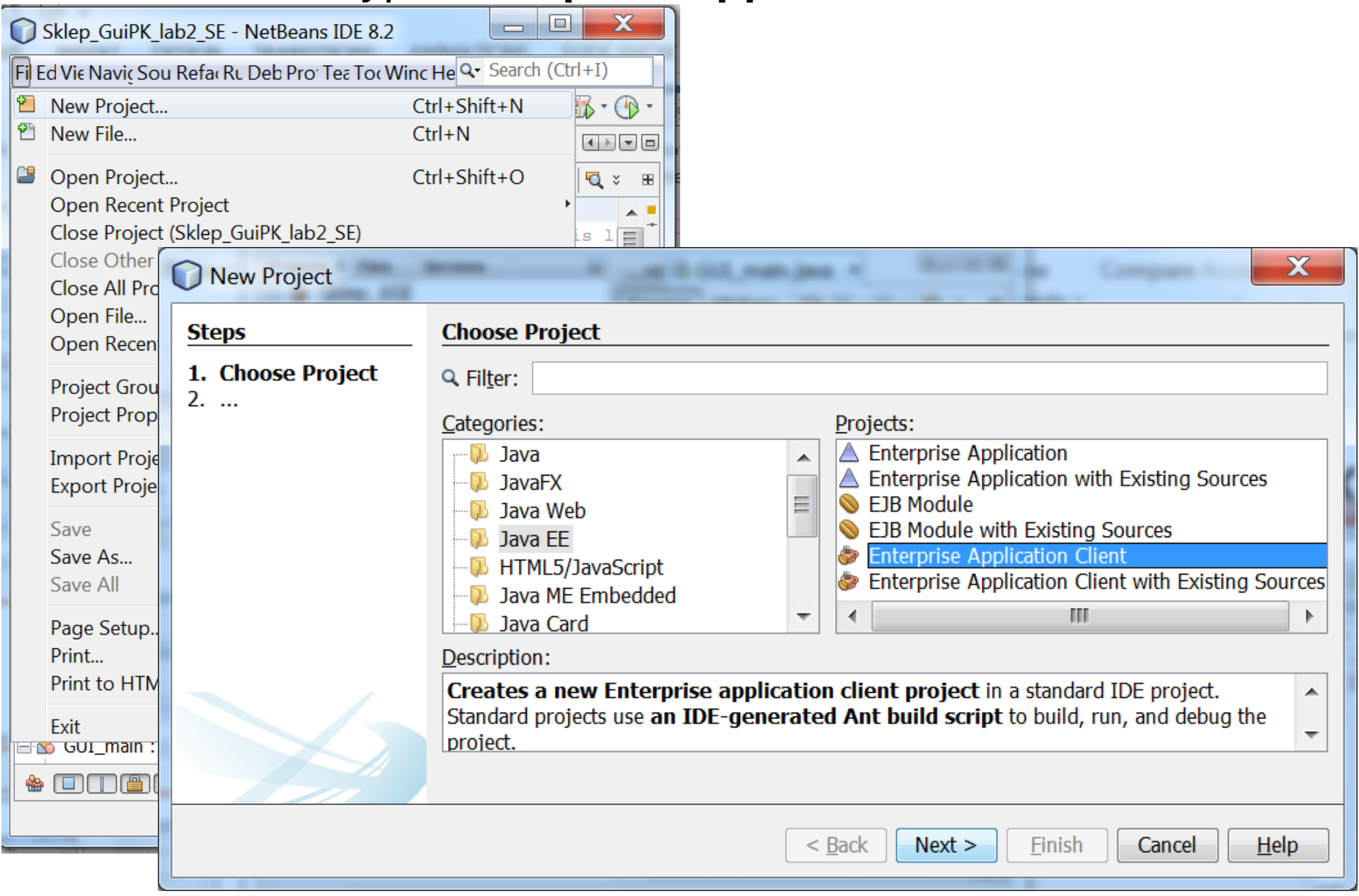


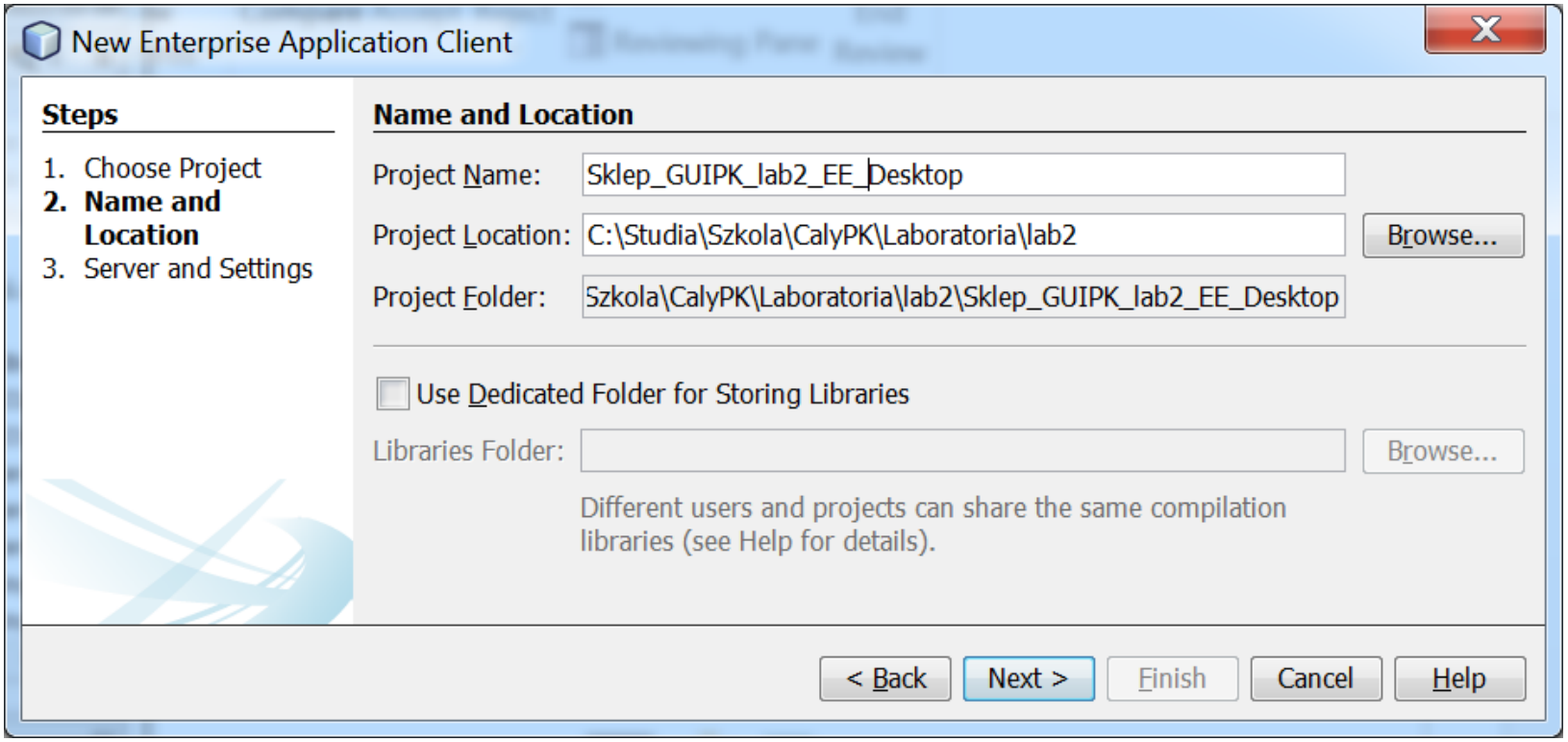


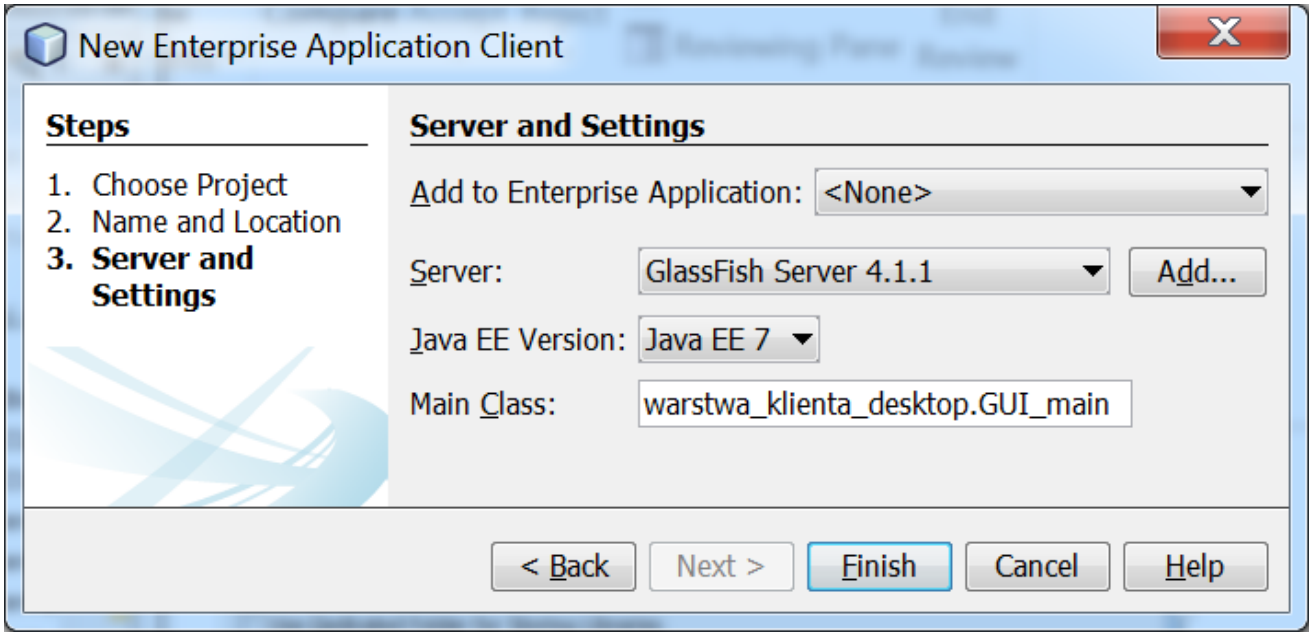
3.7.1. Wykonanie i uruchomienie projektu Java SE o architekturze dwuwarstwowej.

Takie podejście ułatwia wdrożenie deskotopowego klienta aplikacji EE.

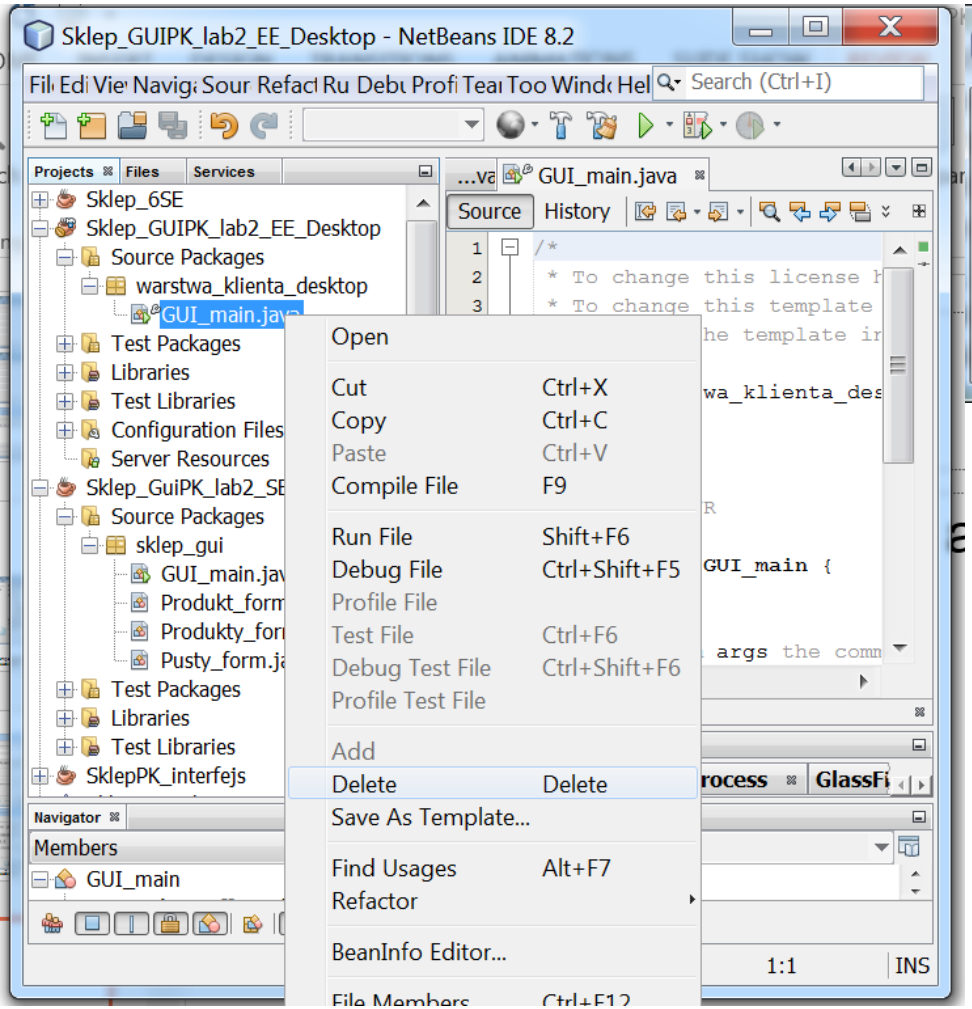
3.7.2. Drugi etap dodania warstwy klienta desktopowego **Java EE** typu **Enterprise Application Client.**

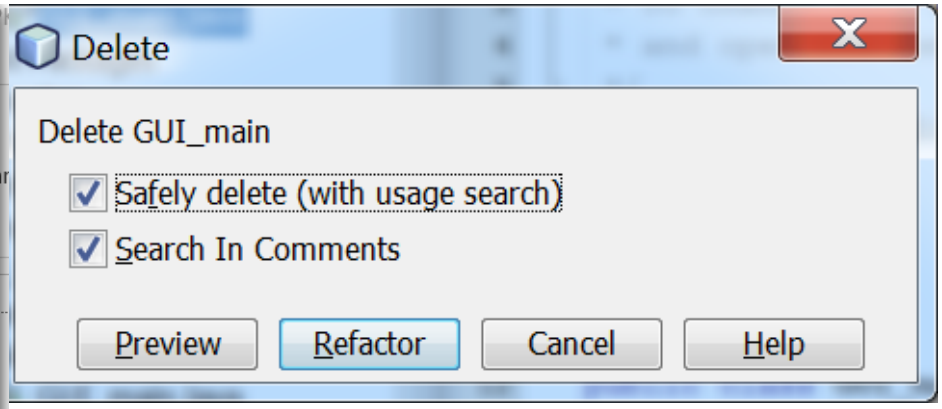



New Enterprise Application Client
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
Name and Location
Project Name: Sklep_GUIPK_lab2_EE_Desktop
Project Location: C:\Studia\Szkola\CalyPK\Laboratoria\lab2
Browse...
Project Folder: Szkola\CalyPK\Laboratoria\lab2\Sklep_GUIPK_lab2_EE_Desktop
Use Dedicated Folder for Storing Libraries
Libraries Folder:
Browse...
Different users and projects can share the same compilation libraries (see Help for details).
< Back
Next >
Finish
Cancel
Help

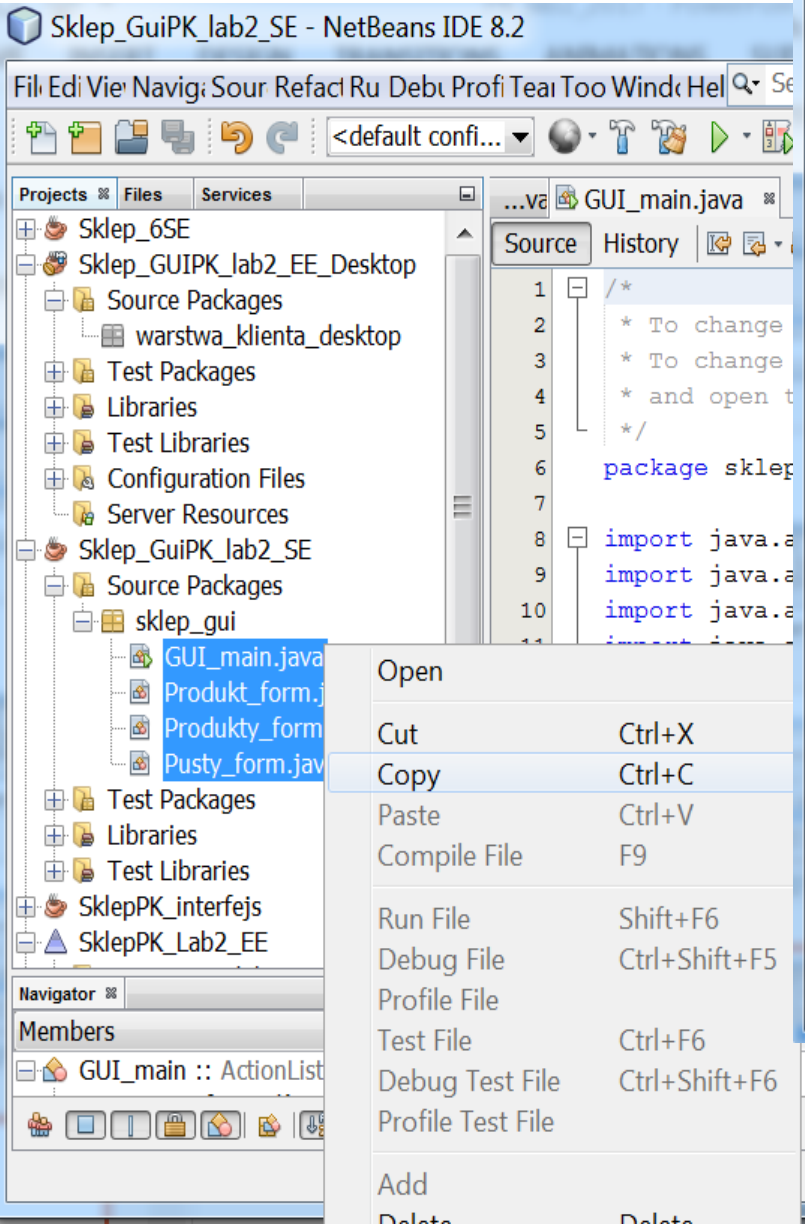

New Enterprise Application Client
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
Server and Settings
Add to Enterprise Application: <None>
Server: GlassFish Server 4.1.1
Add...
Java EE Version: Java EE 7
Main Class: warstwa_klienta_desktop.GUI_main
< Back
Next >
Finish
Cancel
Help

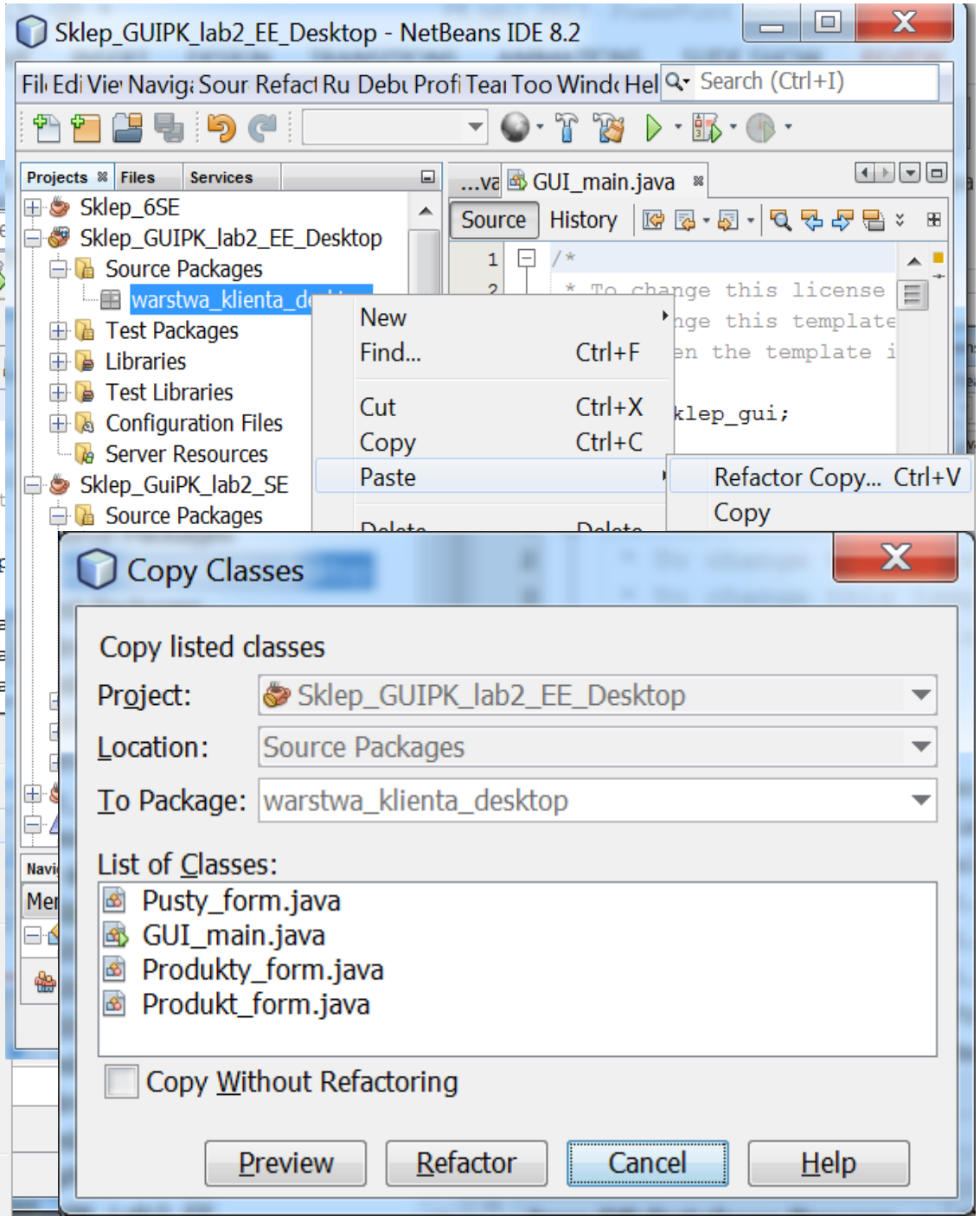
3.7.3. Usunięcie pliku GUI_main



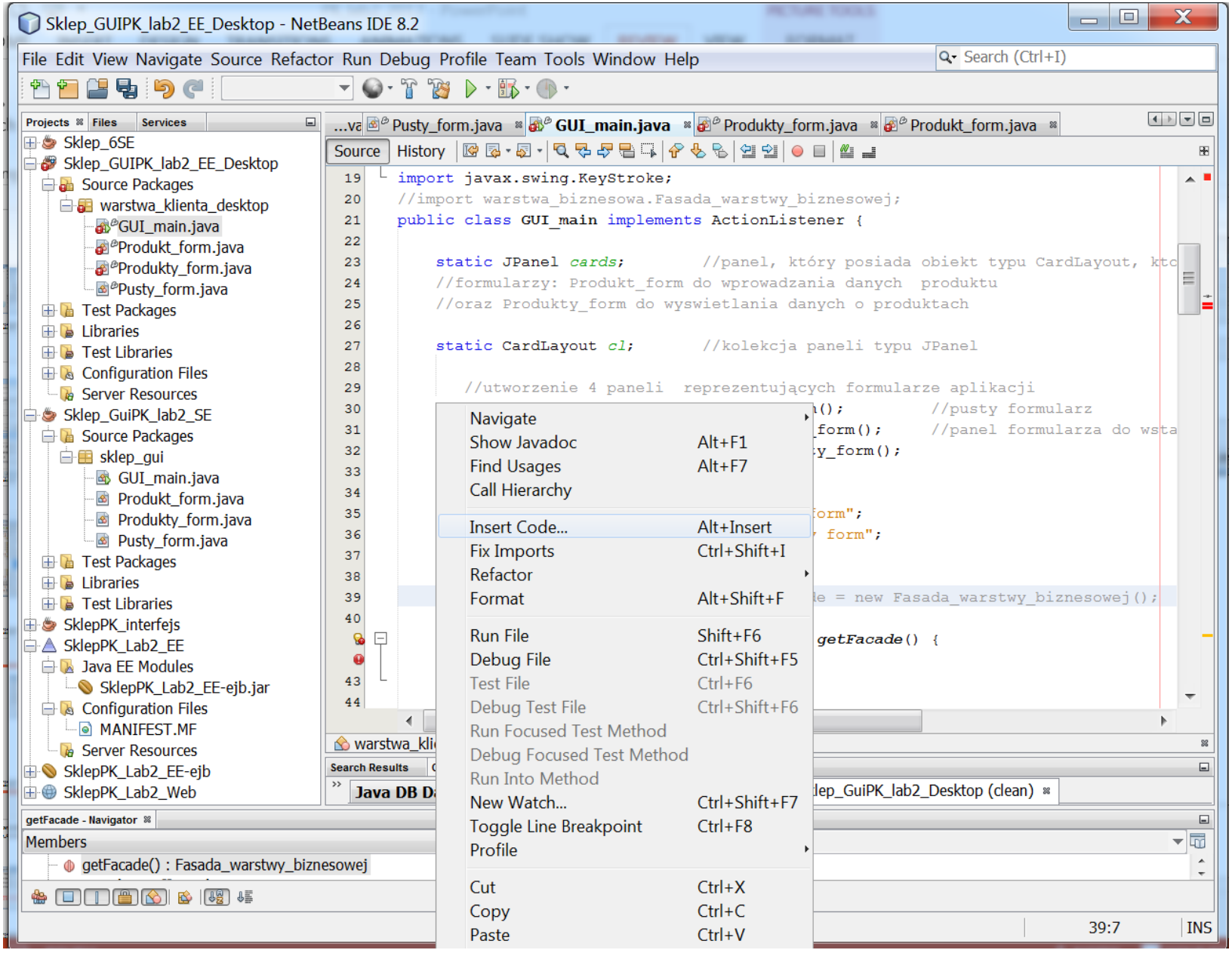


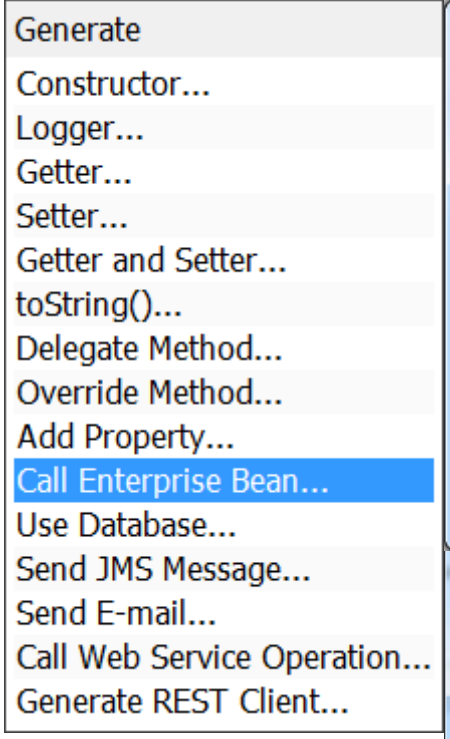
3.7.4. Skopiowanie plików z formularzami z projektu klienta SE: **Sklep_GuiPK_lab2_SE.**

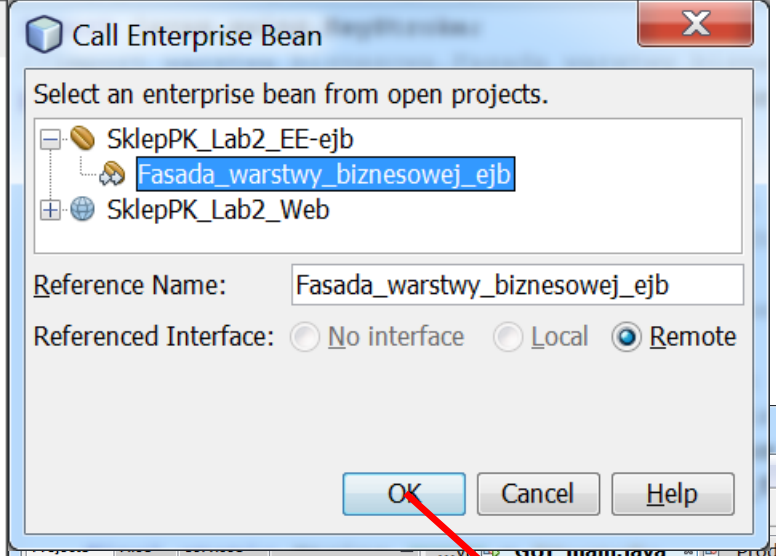


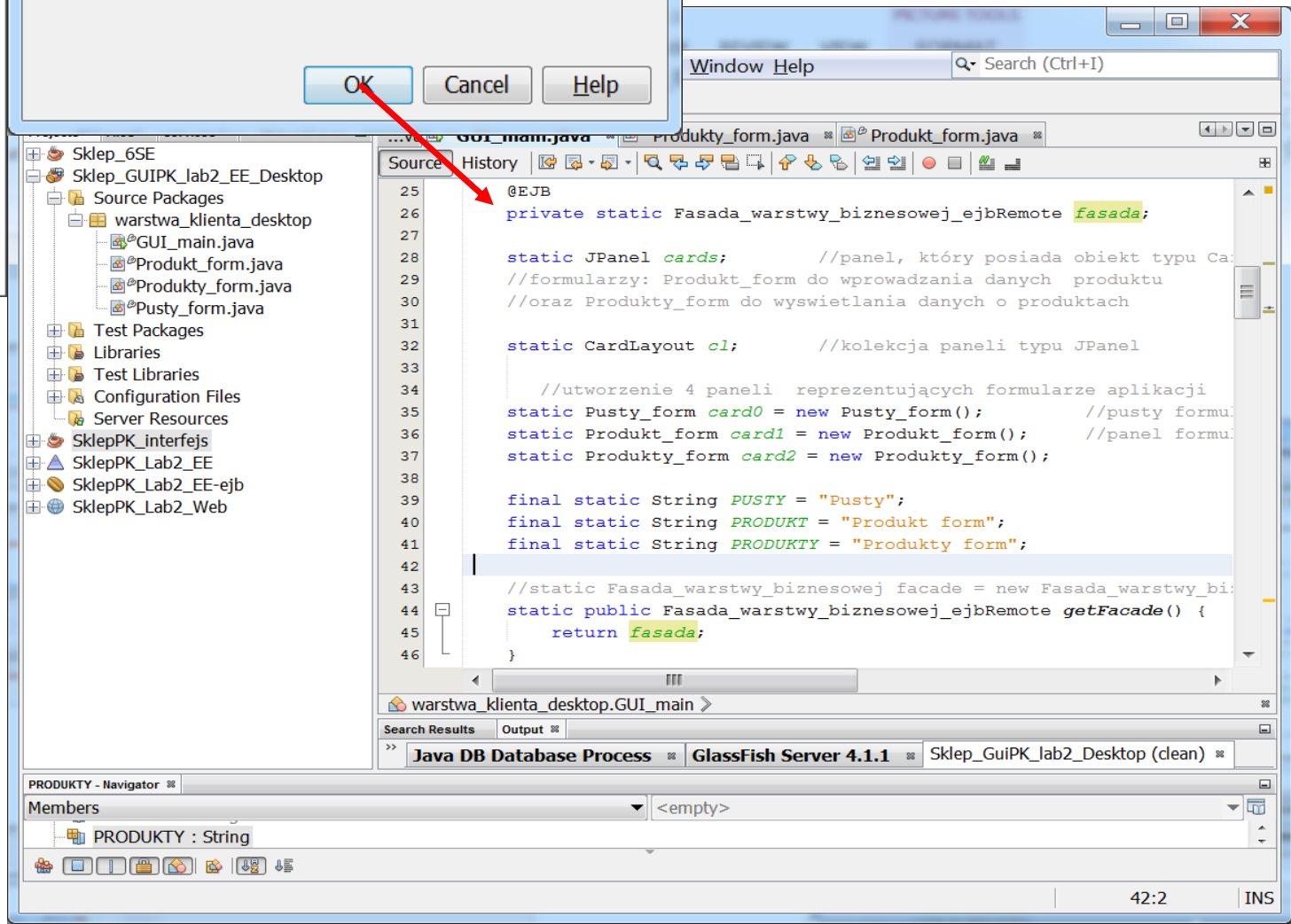


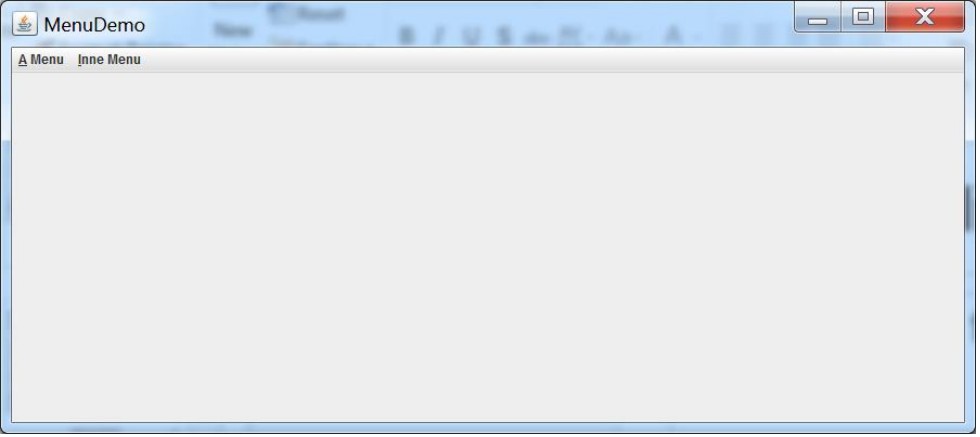
3.7.5. Należy w pliku GUI_main dodać połączenie z komponentem EJB typu Session: **Fasada_warstwy_biznesowej_ejb**, umieszczonym w module EJB

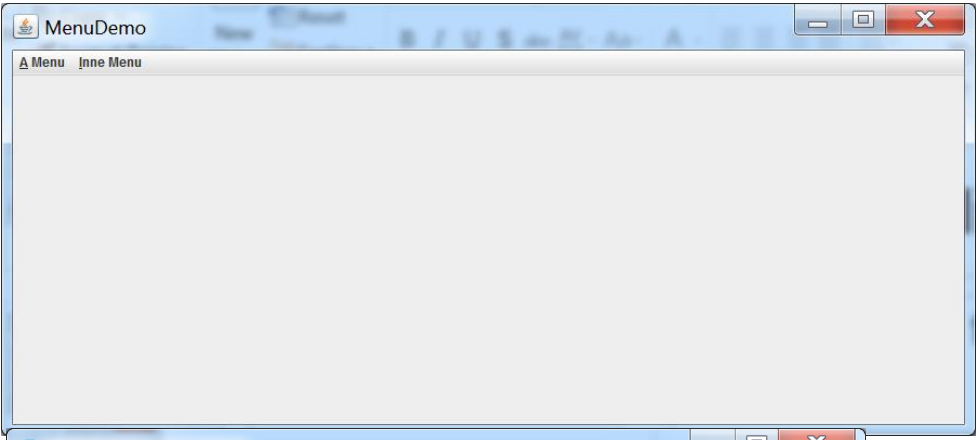


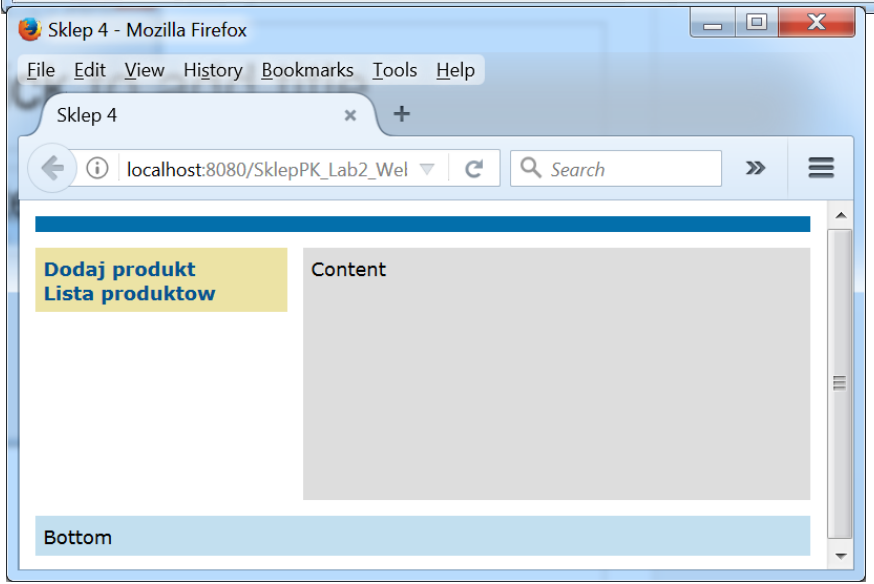

Generate
Constructor...
Logger...
Getter...
Setter...
Getter and Setter...
toString()...
Delegate Method...
Override Method...
Add Property...
Call Enterprise Bean...
Use Database...
Send JMS Message...
Send E-mail...
Call Web Service Operation...
Generate REST Client...


Call Enterprise Bean
Select an enterprise bean from open projects.
SklepPK_Lab2_EE-ejb
Fasada_warstwy_biznesowej_ejb
SklepPK_Lab2_Web
Reference Name:
Fasada_warstwy_biznesowej_ejb
Referenced Interface:
No interface
Local
Remote
OK
Cancel
Help


Window Help
Search (Ctrl+I)
Sklep_6SE
Sklep_GUIPK_lab2_EE_Desktop
Source Packages
warstwa_klienta_desktop
GUI_main.java
Produkt_form.java
Produkty_form.java
Pusty_form.java
Test Packages
Libraries
Test Libraries
Configuration Files
Server Resources
SklepPK_interfejs
SklepPK_Lab2_EE
SklepPK_Lab2_EE-ejb
SklepPK_Lab2_Web
Produkty_form.java
Produkt_form.java
Source
History
@EJB
private static Fasada_warstwy_biznesowej_ejbRemote fasada;
static JPanel cards; //panel, który posiada obiekt typu Ca
//formularzy: Produkt_form do wprowadzania danych produktu
//oraz Produkty_form do wyswietlania danych o produktach
static CardLayout cl; //kolekcja paneli typu JPanel
//utworzenie 4 paneli reprezentujących formularze aplikacji
static Pusty_form card0 = new Pusty_form(); //pusty formu
static Produkt_form card1 = new Produkt_form(); //panel formu
static Produkty_form card2 = new Produkty_form();
final static String PUSTY = "Pusty";
final static String PRODUKT = "Produkt form";
final static String PRODUKTY = "Produkty form";
//static Fasada_warstwy_biznesowej facade = new Fasada_warstwy_bi
static public Fasada_warstwy_biznesowej_ejbRemote getFacade() {
return fasada;
}
warstwa_klienta_desktop.GUI_main
Search Results
Output
Java DB Database Process
GlassFish Server 4.1.1
Sklep_GuiPK_lab2_Desktop (clean)
PRODUKTY - Navigator
Members
<empty>
PRODUKTY : String
42:2
INS







3.8. Uruchomienie projektu.

Należy w podanej kolejności wykonać operacje **Clean and Build** na projektach składowych (w celu łatwiejszej lokalizacji błędów):

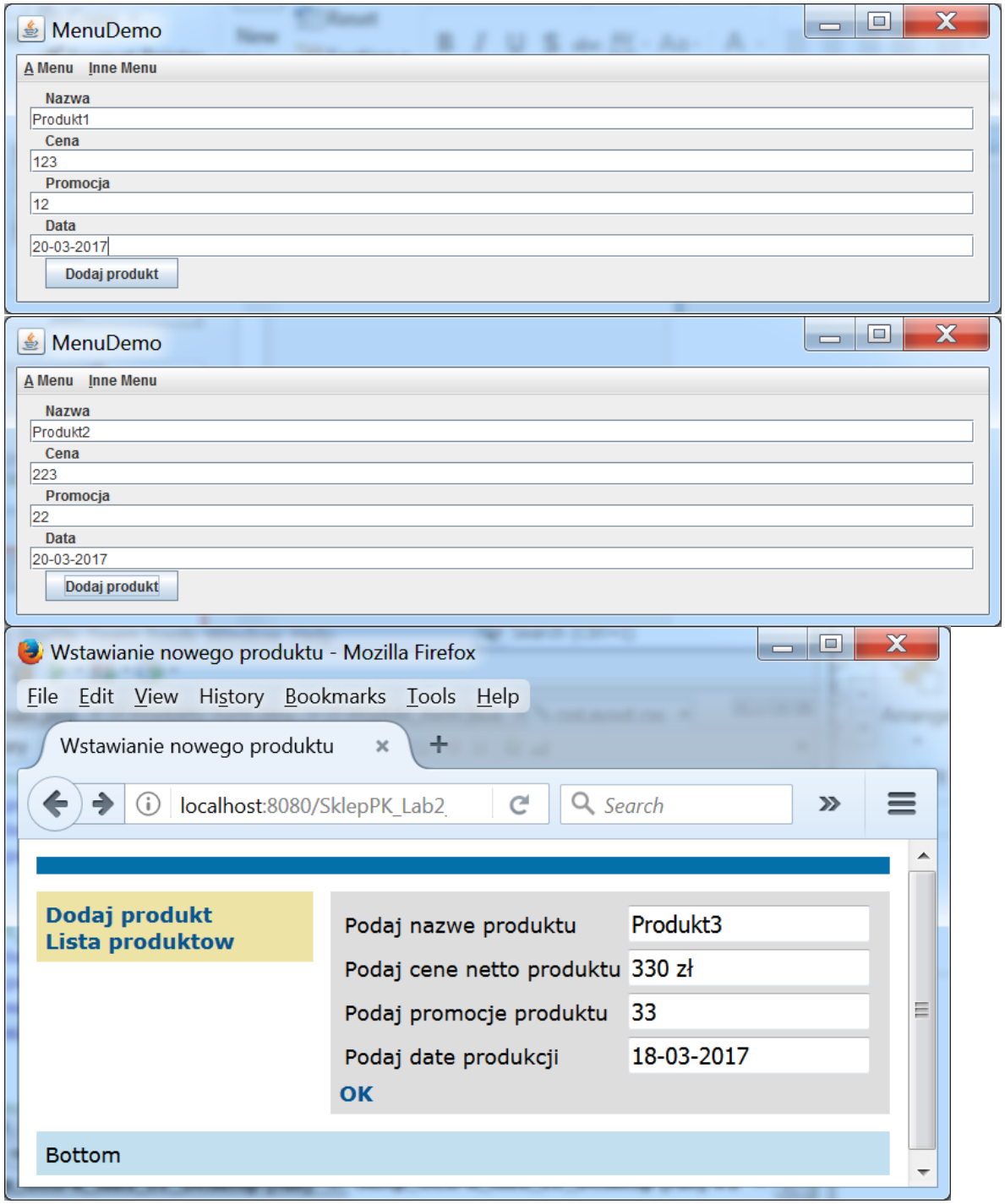
1) Sklep_6SE
2) SklepPK_Lab2_EE-ejb
3) SklepPK_Lab2_Web
4) Sklep_GUIPK_lab2_EE_Desktop
5) SklepPK_Lab2_EE

Następnie, należy wykonać operację **Deploy** na projekcie **SklepPK_Lab2_EE**.

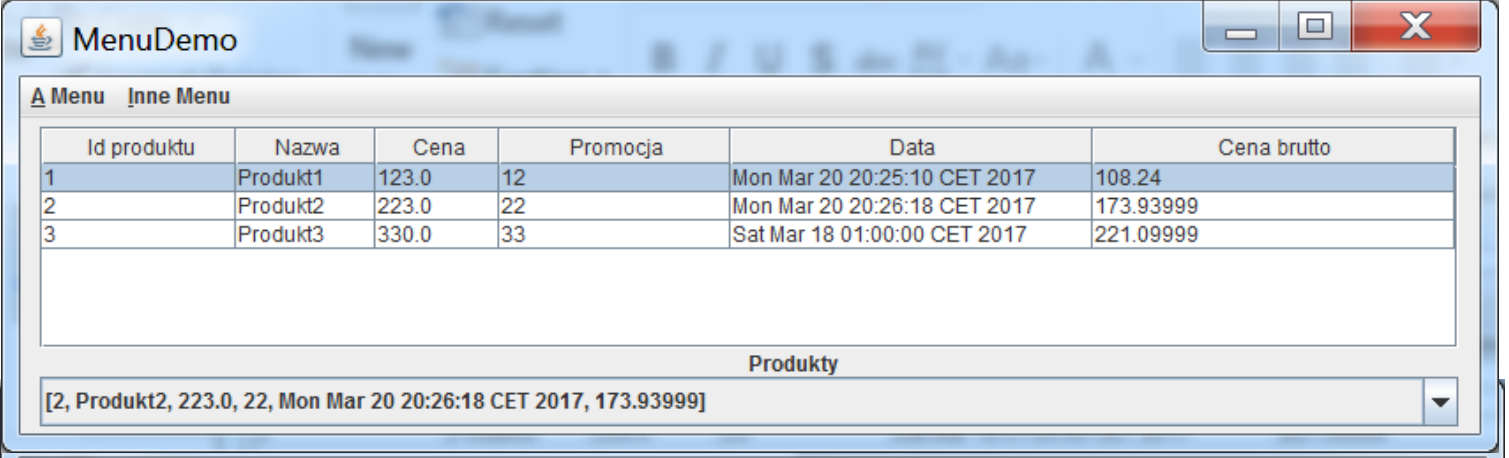
Teraz można uruchomić dowolną liczbę aplikacji klienckich za pomocą operacji **Run**:

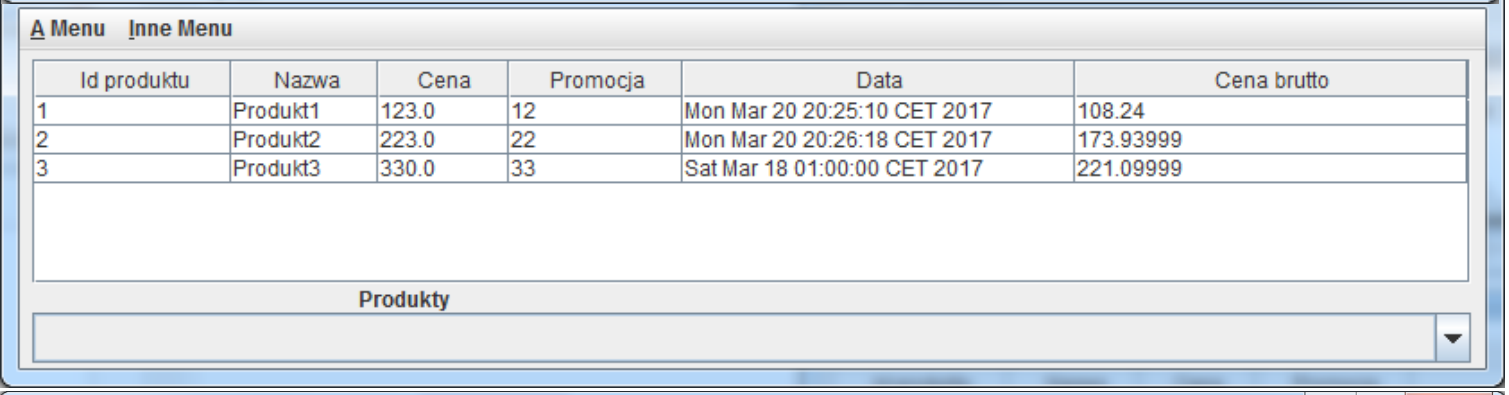
1) SklepPK_Lab2_Web
2) Sklep_GUIPK_lab2_EE_Desktop

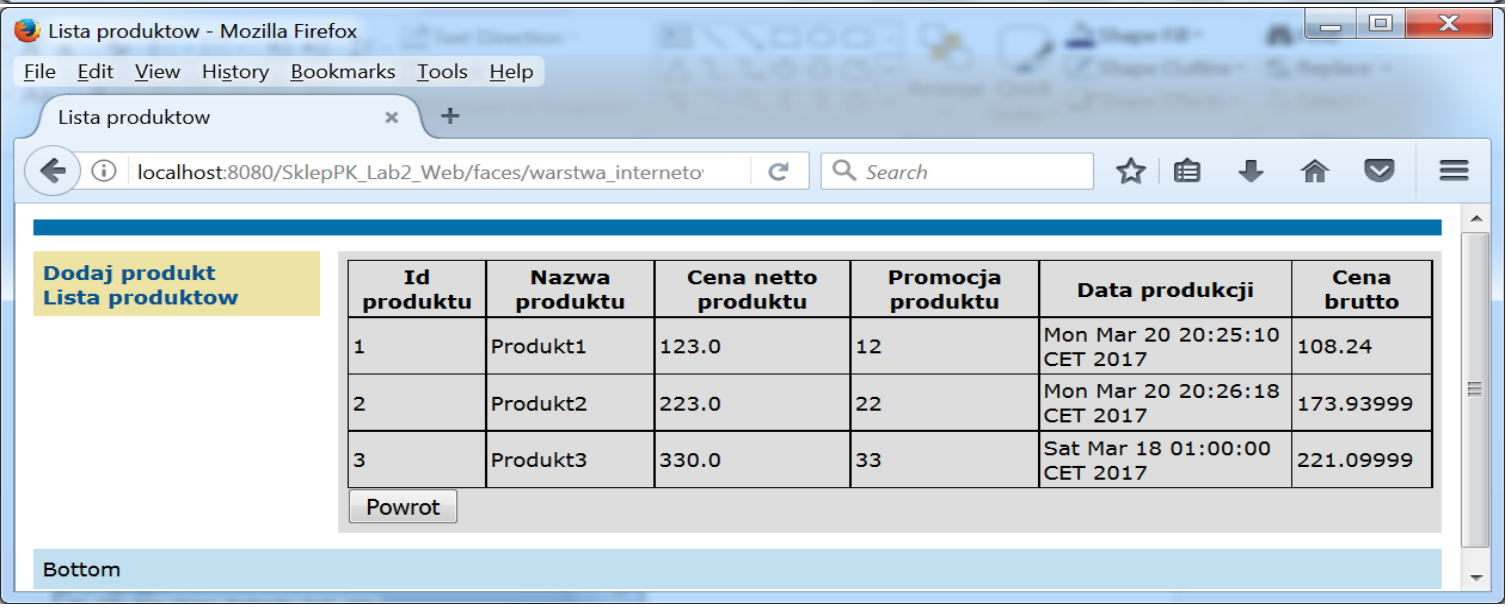
W przykładzie uruchomiono dwie instancje aplikacji desktopowej i jedną internetową.



3.8.1. Proces wprowadzania danych w uruchomionych aplikacjach klienckich







3.8.2. Każda z aplikacji klienckich korzysta z danych umieszczonych w tym samym **komponencie EJB typu Session – rodzaj Stateless**.