Zaawansowane aplikacje internetowe - laboratorium

Hibernate.

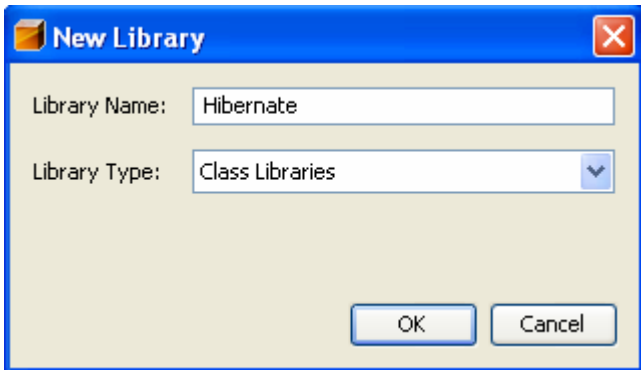
Do wykonania ćwiczeń potrzebne jest zintegrowane środowisko programistyczne NetBeans IDE 5.5 wraz z serwerem Sun Java System Application Server Platform Edition 9 (do pobrania z http://www.netbeans.org/downloads/index.html jako Java EE 5 Tools Bundle) oraz środowisko J2SE w wersji 1.5 Update 1 (lub wyższej) wymagane do instalacji NetBeans. Instalując Java EE 5 Tools Bundle należy zainstalować wszystkie składniki wraz z zawartym w pakiecie serwerem aplikacji (wybór opcji "Install the bundled Java EE SDK" na jednym z ekranów instalatora).

Ćwiczenie 1

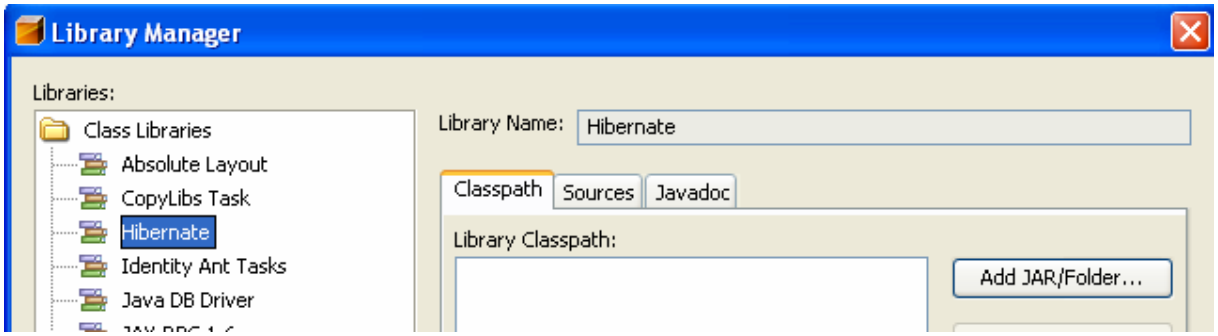
Celem ćwiczenia jest pobranie biblioteki Hibernate i jej instalacja w środowisku NetBeans. Ze względu na fakt, że Hibernate nie jest standardową biblioteką, konieczne jest jej ręczne pobranie i instalacja w formie biblioteki lub wtyczki czy rozszerzenia dla środowiska IDE. W ćwiczeniu Hibernate zostanie zainstalowany w formie biblioteki, rozumianej jako nazwany zbiór archiwów JAR.

Kroki ćwiczenia:

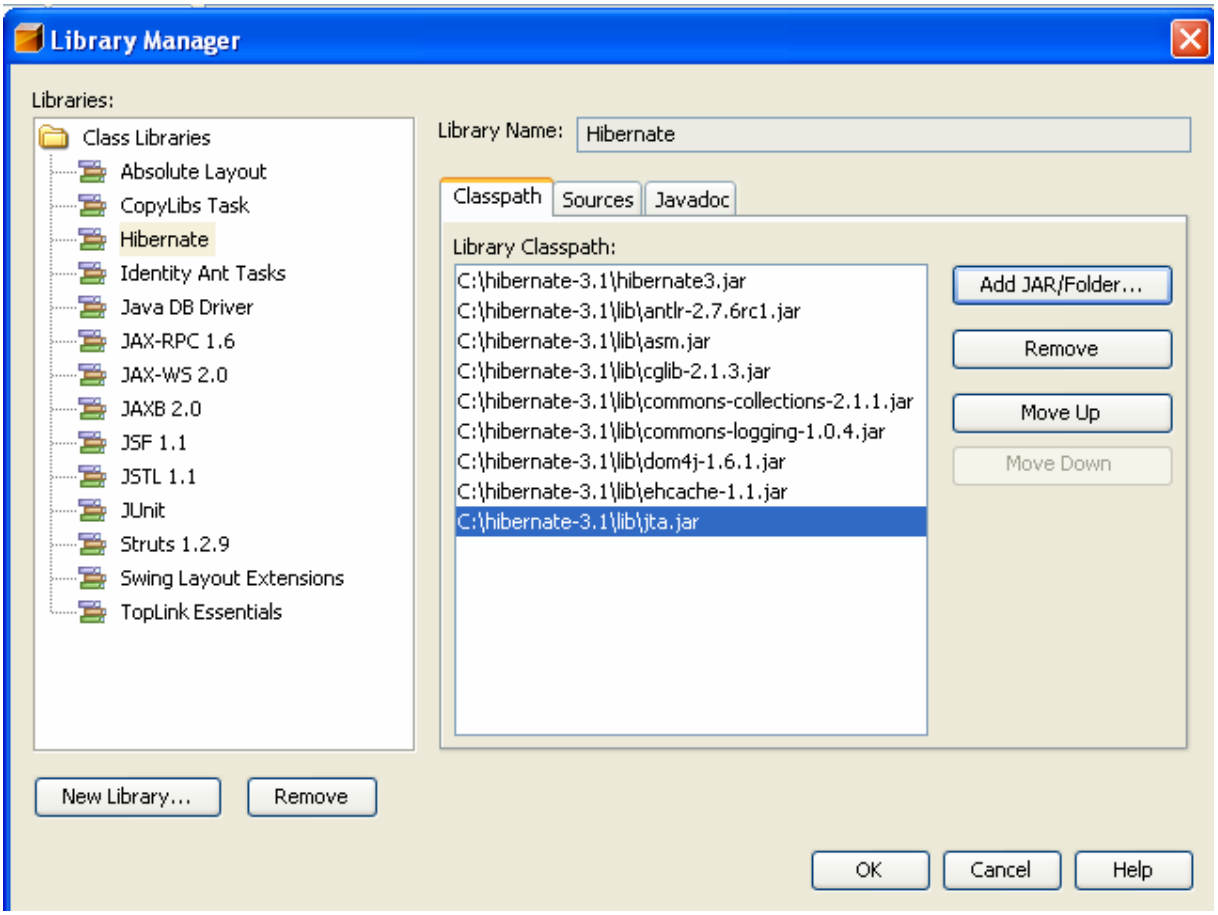
1. Pobranie biblioteki Hibernate z Internetu
   a) Za pomocą przeglądarki internetowej przejdź na stronę http://www.hibernate.org/. Z menu po lewej wybierz link Download. Na kolejnej stronie z sekcji Binary Releases wybierz najnowszą wersję Hibernate Core o statusie Production. Na kolejnej stronie wybierz platform-independent zip. Zapisz plik na dysku pobierając go z jednego ze wskazanych dostępnych serwerów.
   b) Rozpakuj pobrane archiwum ZIP na dysku lokalnym. W dalszej części ćwiczenia będzie przyjęte, że zawartość archiwum została wypakowana jako folder `C:\hibernate-3.1` (nazwa katalogu zawartego w archiwum ZIP zależy od wersji biblioteki Hibernate i może być inna).
   c) Obejrzyj zawartość folderu z biblioteką Hibernate. Główny katalog zawiera archiwum JAR z klasami Hibernate (hibernate3.jar dla wersji 3.x Hibernate). Podkatalog `doc` zawiera dokumentację, a podkatalog `lib` biblioteki, z których korzysta Hibernate, w odpowiednich wersjach. Część z tych bibliotek jest wymagana, a część ma charakter opcjonalny.
2. Instalacja biblioteki Hibernate w środowisku NetBeans
   a) Uruchom narzędzie NetBeans IDE 5.5.
   b) Z menu głównego wybierz Tools→Library Manager. Kliknij przycisk New Library.
   c) Jako nazwę biblioteki wprowadź „Hibernate” i kliknij przycisk OK.



d) Zaznacz w lewym panelu bibliotekę Hibernate i kliknij przycisk Add JAR/Folder....



e) Powtarzając operację Add JAR/Folder... dodaj kolejno archiwum JAR `hibernate` i zawarte w podkatalogu `lib` dystrybucji Hibernate wymagane archiwa JAR `antlr`, `asm`, `cglib`, `commons-collections`, `commons-logging`, `dom4j`, `ehcache` i `jta`.



f) Kliknij przycisk OK.

Ćwiczenie 2

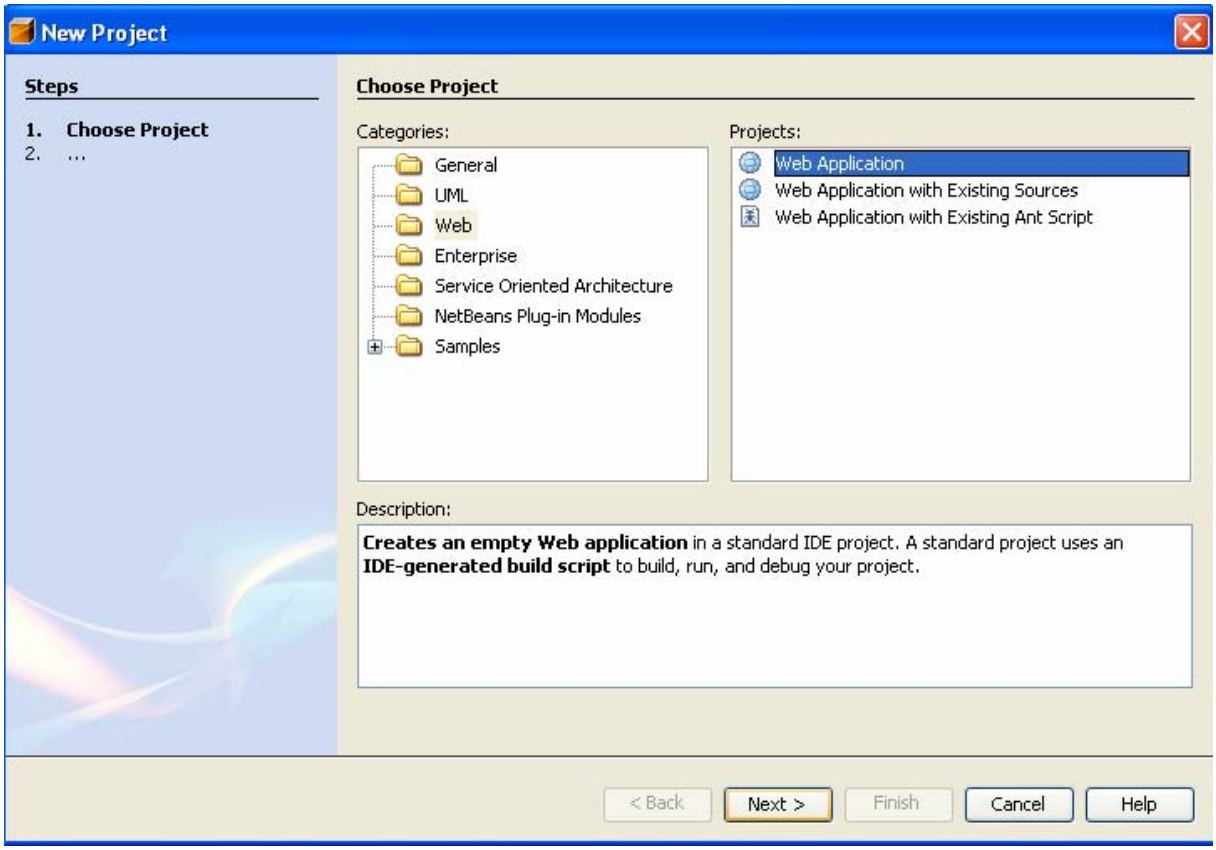
Celem ćwiczenia jest przygotowanie prostej aplikacji JSF do przeglądania i edycji katalogu produktów, realizującej odczyt i zapis danych z/do bazy danych poprzez bibliotekę Hibernate. Ćwiczenie wykorzystuje serwer bazy danych Derby (Java DB), instalowany jako składnik Java EE 5 Tools Bundle.

Kroki ćwiczenia:

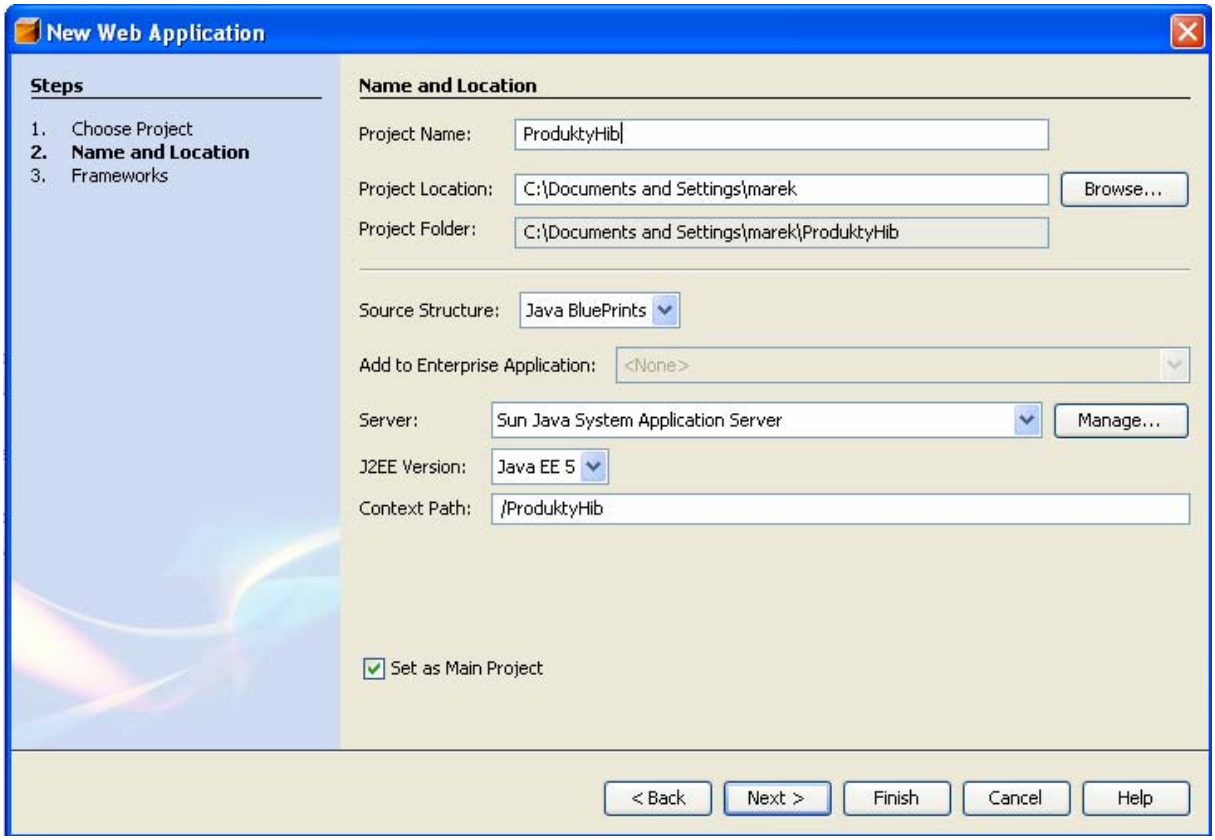
1. Utworzenie nowego projektu i dodanie do niego wymaganych bibliotek.

a) Uruchom narzędzie NetBeans IDE 5.5

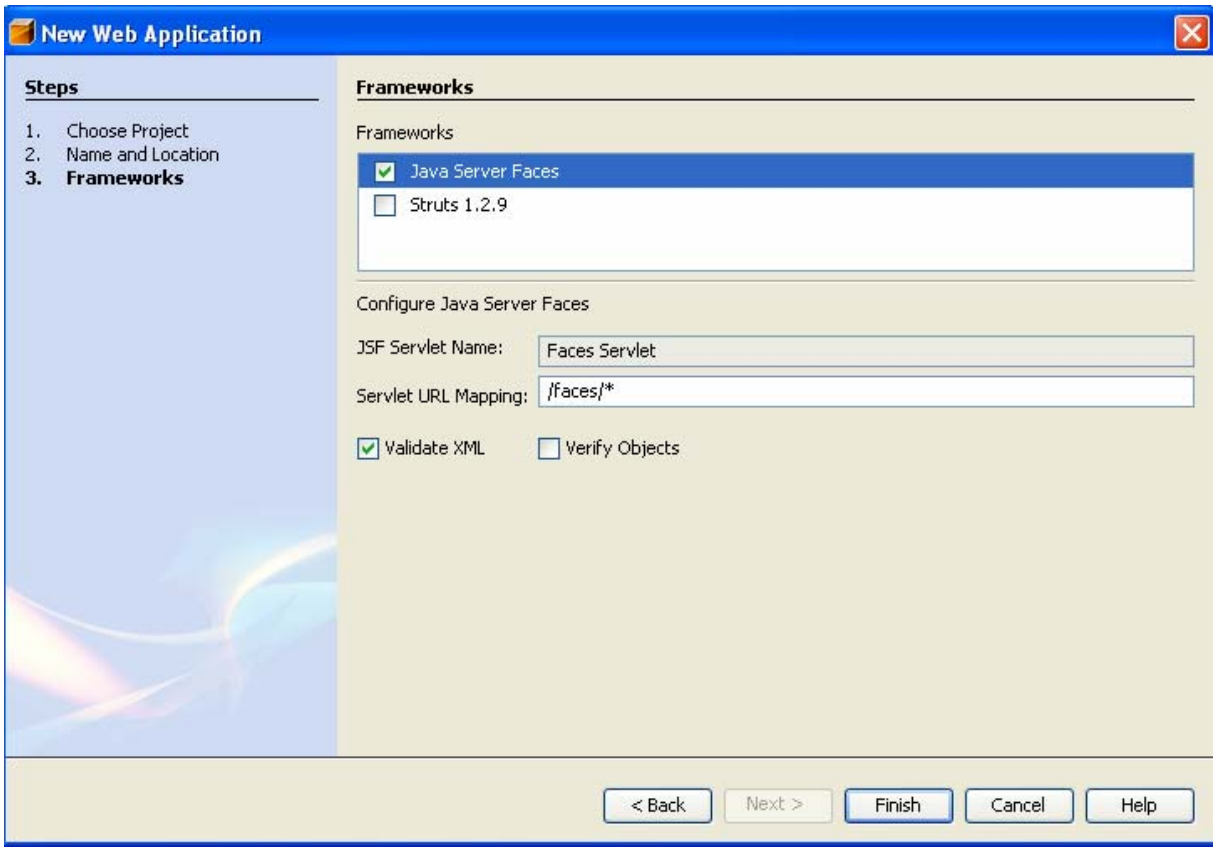
b) Z menu głównego wybierz File→New Project. Wybierz kategorię Web i typ projektu Web Application. Kliknij przycisk Next >.



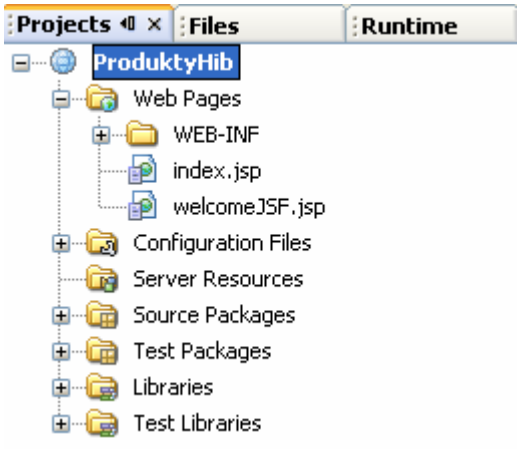
c) Podaj nazwę projektu, „ProduktyHib”. Zwróć uwagę, że wraz zmianą nazwy projektu zmienia się Context Path czyli katalog wirtualny na serwerze WWW, który będzie prowadził do aplikacji. W polu Project Location powinien być wskazany katalog, w którym masz prawo zapisu. Jako Server powinien być wybrany Sun Java System Application Server a jako J2EE Version – Java EE 5. Kliknij przycisk Next >.



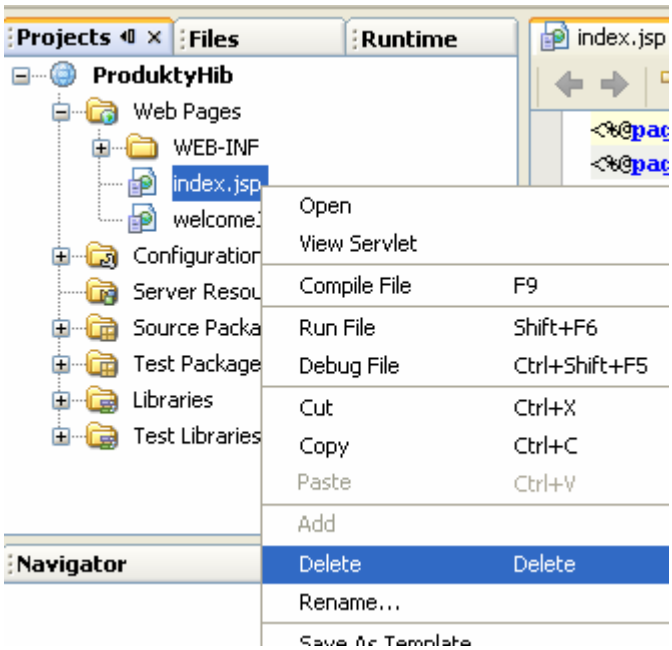
d) Zaznacz Java Server Faces w panelu Frameworks. Pozostaw domyślne opcje konfiguracji aplikacji JSF, które pojawią się po zaznaczeniu pola wyboru Java Server Faces. Kliknij przycisk Finish.



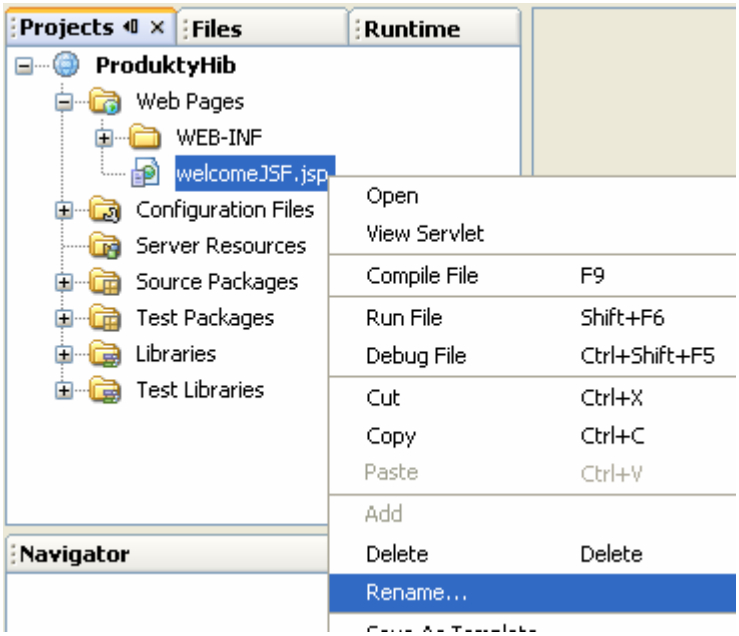
e) Kreator tworzy aplikację zawierającą jedną stronę JSP i jedna stronę JSF. Rozwiń drzewo projektu w panelu Projects.



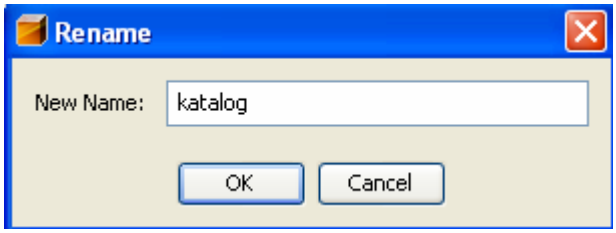
f) Usuń z projektu stronę `index.jsp` wywołując prawym klawiszem myszy menu kontekstowe dla pliku w drzewie projektu i wybierając z niego opcję Delete.



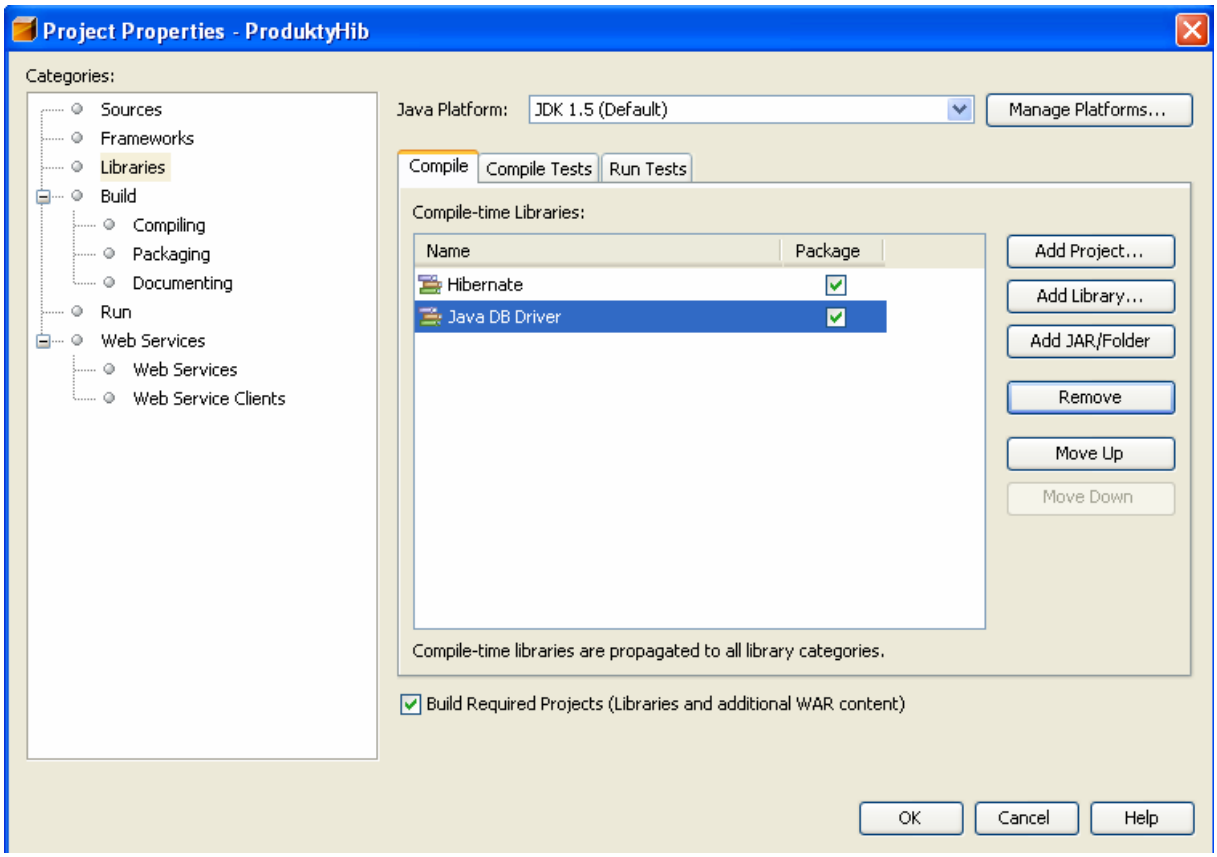
g) Zmień nazwę strony `welcomeJSF.jsp` na `katalog.jsp` wywołując prawym klawiszem myszy menu kontekstowe dla pliku w drzewie projektu i wybierając z niego opcję Rename.



W oknie dialogowym Rename podaj nazwę bez rozszerzenia (`katalog`).



h) Dodaj do projektu bibliotekę Hibernate utworzoną w Ćwiczeniu 1 i bibliotekę sterownika JDBC dla bazy danych Derby (Java DB). W tym celu kliknij prawym przyciskiem myszy na ikonie projektu w drzewie projektów i z menu kontekstowego wybierz Properties. Następnie wybierz kategorię Libraries i klikając przycisk Add Library... wybierz i dodaj do projektu biblioteki Hibernate i Java DB Driver.



i) Kliknij przycisk OK.

2. Utworzenie klasy trwałej dla Hibernate.

a) Kliknij prawym przyciskiem myszy na ikonie projektu w drzewie projektów i z menu kontekstowego wybierz New→File/Folder. Kliknij przycisk Next >.

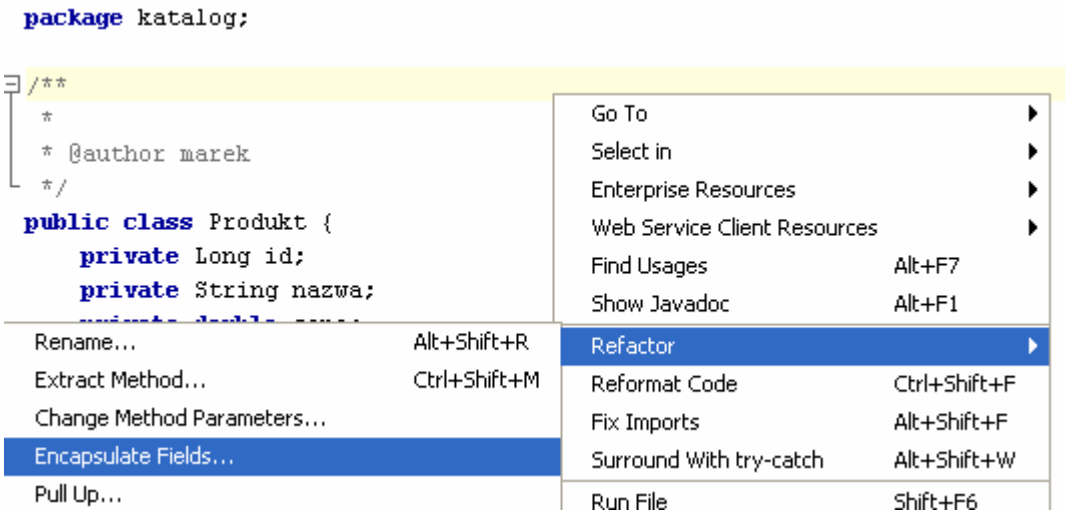
b) Następnie wybierz kategorię Java Classes i typ pliku Java Class. Kliknij przycisk Next >.

c) Jako nazwę klasy podaj „Produkt”, a jako nazwę pakietu „katalog”. W pozostałych polach pozostaw wartości zaproponowane przez kreator. Kliknij przycisk Finish.

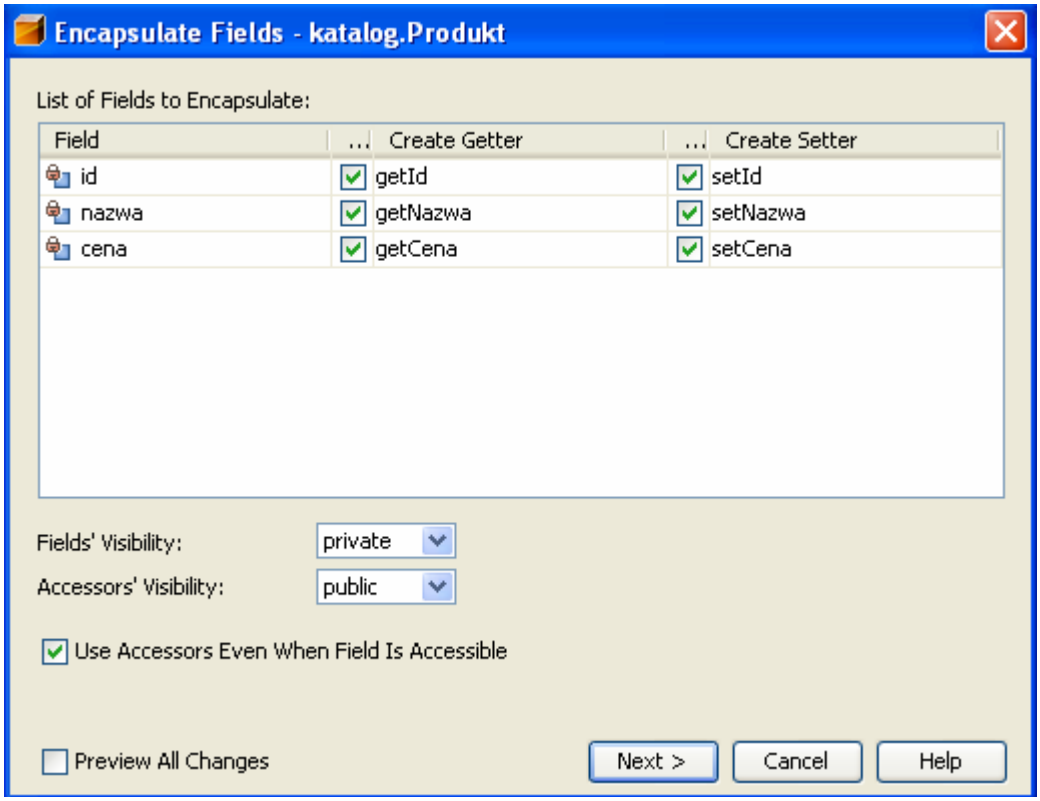
d) Dodaj w klasie Produkt trzy prywatne pola reprezentujące identyfikator, nazwę i cenę produktu:

```
...
public class Produkt {
    private Long id;
    private String nazwa;
    private double cena;
...
```

e) Dodaj metody set/get dla wszystkich pól klasy, udostępniając je jako właściwości JavaBean. Skorzystaj z kreatora, wywołując prawym klawiszem myszy menu kontekstowe z poziomu kodu klasy i wybierając opcję Refactor→Encapsulate fields.



Upewnij się, że pola wyboru dla wszystkich metod set/get są zaznaczone. Odznacz pole Preview all changes, aby zmiany nie wymagały potwierdzenia i kliknij przycisk Next >.



3. Definicja odwzorowania klasy trwałej na tabelę w bazie danych.

a) Kliknij prawym przyciskiem myszy na ikonie projektu w drzewie projektów i z menu kontekstowego wybierz New→File/Folder. Kliknij przycisk Next >.

b) Następnie wybierz kategorię XML i typ pliku XML Document. Kliknij przycisk Next >.

c) Jako nazwę klasy pliku „Produkt.hbm”, a jako nazwę folderu - folder zawierający źródłowy plik klasy Produkt: „src\java\katalog” (nazwę folderu możesz wprowadzić

ręcznie lub wybrać kreatorem wywoływanym przyciskiem Browse...). Kliknij przycisk Next.

d) W kolejnym oknie wybierz opcję Well-formed Document (powinna być wybrana domyślnie) i kliknij przycisk Finish.

e) Zastąp treść utworzonego pliku `Produkt.hbm.xml` poniższą definicją odwzorowania:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="katalog">
    <class name="Produkt" table="PRODUKTY">
  <id name="id" type="long" column="produkt_id">
    <generator class="native" />
  </id>
  <property name="nazwa" type="string" not-null="true" />
  <property name="cena" type="double" not-null="true" />
</class>
</hibernate-mapping>
```

Powyższy kod odwzorowuje klasę Produkt na tabelę PRODUKTY zawierającą kolumny: produkt_id, nazwa i cena. Kluczem głównym tabeli jest produkt_id. Wartości klucza mają być generowane domyślnym natywnym sposobem dla wykorzystywanej bazy danych.

f) Zapisz wszystkie zmiany (File→Save All lub ikona w pasku narzędzi).

4. Konfiguracja Hibernate.

a) Kliknij prawym przyciskiem myszy na ikonie projektu w drzewie projektów i z menu kontekstowego wybierz New→File/Folder. Kliknij przycisk Next >.

b) Następnie wybierz kategorię XML i typ pliku XML Document. Kliknij przycisk Next >.

c) Jako nazwę klasy pliku „hibernate.cfg", a jako nazwę folderu – główny folder zawierający podkatalog pakietu ze źródłowym plikiem klasy Produkt: „src\java" (nazwę folderu możesz wprowadzić ręcznie lub wybrać kreatorem wywoływanym przyciskiem Browse...). Kliknij przycisk Next.

d) W kolejnym oknie wybierz opcję Well-formed Document (powinna być wybrana domyślnie) i kliknij przycisk Finish.

e) Zastąp treść utworzonego pliku `hibernate.cfg.xml` poniższą zawartością:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
  <property name="hibernate.connection.datasource">jdbc/sample</property>
  <property name="hibernate.connection.username">app</property>
  <property name="hibernate.connection.password">app</property>
  <property name="dialect">org.hibernate.dialect.DerbyDialect</property>
  <property name="hibernate.hbm2ddl.auto">update</property>
  <mapping resource="katalog/Produkt.hbm.xml"/>
</session-factory>
</hibernate-configuration>
```

Powyższa konfiguracja wykorzystuje źródło danych jdbc/sample automatycznie dostępne na serwerze Sun Java System Application Server Platform Edition 9 instalowanym jako składnik Java EE 5 Tools Bundle, reprezentujące wbudowany serwer bazy danych Derby (Java DB). Wskazany użytkownik w bazie danych, to automatycznie dostępny użytkownik app z hasłem „app". Wartość update właściwości hibernate.hbm2ddl.auto oznacza, że Hibernate ma przy uruchomieniu aplikacji utworzyć schemat w bazie danych jeśli nie będzie istniał lub go uaktualnić jeśli nie będzie odpowiadał zawartości plików odwzorowań. Pozostałe dwa wiersze pliku konfiguracyjnego to wybór dialektu odpowiedniego dla wykorzystywanej bazy danych Derby i wskazanie lokalizacji jedynego w aplikacji pliku z odwzorowaniem.

f) Zapisz wszystkie zmiany (File→Save All lub ikona w pasku narzędzi).

5. Utworzenie klasy pomocniczej udostępniającej obiekt SessionFactory.

a) Kliknij prawym przyciskiem myszy na ikonie projektu w drzewie projektów i z menu kontekstowego wybierz New→File/Folder. Kliknij przycisk Next >.

b) Następnie wybierz kategorię Java Classes i typ pliku Java Class. Kliknij przycisk Next >.

c) Jako nazwę klasy podaj „HibernateUtil", a jako nazwę pakietu wybierz z listy „katalog". W pozostałych polach pozostaw wartości zaproponowane przez kreator. Kliknij przycisk Finish.

d) Zastąp kod klasy HibernateUtil poniższą treścią:

```
package katalog;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {

private static final SessionFactory sessionFactory;
  static {
    try {
      sessionFactory = new
                       Configuration().configure().buildSessionFactory();
    } catch (Throwable ex) {
        System.err.println("Utworzenie SessionFactory nieudane: " + ex);
        throw new ExceptionInInitializerError(ex);
      }
  }
  public static SessionFactory getSessionFactory() {
    return sessionFactory;
  }
}
```

e) Zapisz wszystkie zmiany (File→Save All lub ikona w pasku narzędzi).

6. Dodanie na stronie JSF `katalog.jsp` formularza do wprowadzania nowych produktów.

a) Przejdź do edycji pliku `katalog.jsp`.

b) Zmień tytuł strony z „JSP Page" na „Katalog produktów".

c) Zastąp dotychczasową zawartość elementu <f:view></f:view> przez poniższy kod formularza:

```
        <f:view>
            <h:form>
                Nazwa: <h:inputText value="#{prodBean.nazwa}"
                id="username"/>
                Cena: <h:inputText value="#{prodBean.cena}"
                id="password"/>
                <h:commandButton value="Dodaj"
                id="submitButton"
                action="#{prodBean.dodaj}"/>
            </h:form>
        </f:view>
```

Formularz odwołuje się do komponentu backing bean o nazwie katalogBean, który zostanie skonfigurowany w kolejnym kroku ćwiczenia.

7. Utworzenie klasy komponentu backing bean dla strony JSF `katalog.jsp` i jego konfiguracja jako zarządzanego komponentu JavaBean (managed bean).

a) Kliknij prawym przyciskiem myszy na ikonie projektu w drzewie projektów i z menu kontekstowego wybierz New→File/Folder. Kliknij przycisk Next >.

b) Następnie wybierz kategorię Java Classes i typ pliku Java Class. Kliknij przycisk Next >.

c) Jako nazwę klasy podaj „Katalog”, a jako nazwę pakietu podaj „katalog”. W pozostałych polach pozostaw wartości zaproponowane przez kreator. Kliknij przycisk Finish.

d) Dodaj w klasie Katalog dwa pola do zapamiętania wartości wprowadzonych do pól formularza:

```
    private String nazwa;
    private double cena;
```

e) Dodaj w klasie Katalog metody set/get dla obu pól korzystając z kreatora Refactor→Encapsulate fields (analogicznie do punktu 2 e) ćwiczenia).

f) Dodaj w klasie Katalog metodę pomocniczą służącą do wstawiania do tabeli bazy danych nowego produktu:

```
    public void addProdukt(String n, double c) {
        SessionFactory sf = HibernateUtil.getSessionFactory();
        Session s = sf.openSession();
        Transaction tx = s.beginTransaction();
        Produkt p = new Produkt();
        p.setNazwa(n);
        p.setCena(c);
        s.save(p);
        tx.commit();
        s.close();
    }
```

g) Dodaj w pliku `Katalog.java` po deklaracji pakietu dyrektywy import importujące klasy Hibernate wykorzystywane w metodzie addProdukt():

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
```

```
import org.hibernate.Transaction;
```

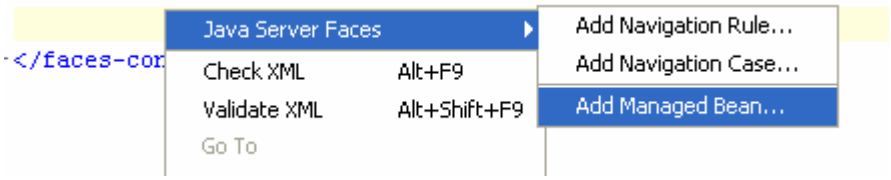
h) Dodaj w klasie Katalog metodę wskazaną na stronie JSF jako metodę obsługi zdarzenia naciśnięcia przycisku:

```
public String dodaj()
{
    addProdukt(nazwa, cena);
    nazwa = null;
    cena = 0.0;
    return null;
}
```

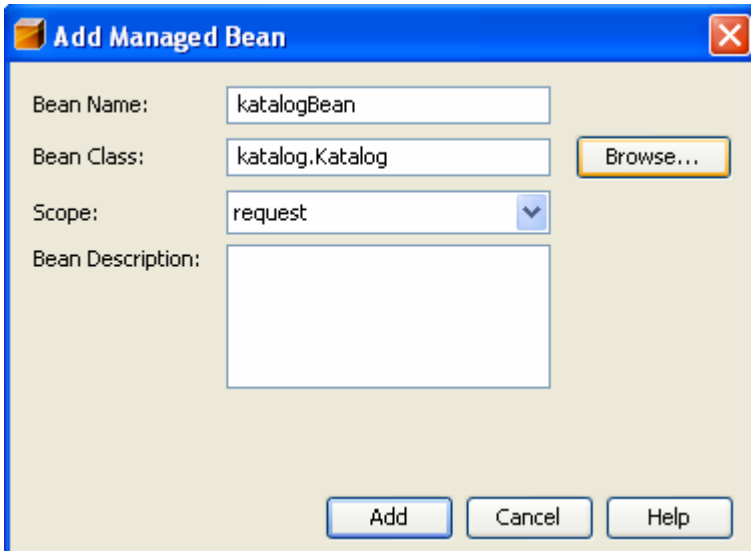
Działanie metody rozpoczyna się od wstawienia do bazy danych produktu o nazwie i cenie wprowadzonych przez użytkownika w formularzu i zapamiętanych we właściwościach komponentu. Następnie zawartość tych właściwości jest resetowana, aby przy ponownym wyświetleniu strony formularz był pusty. Na koniec, metoda zwraca null, co spowoduje ponowne wyświetlenie tej samej strony.

i) Przejdź do edycji pliku konfiguracyjnego aplikacji JSF `faces-config.xml` (dostępnego w gałęzi Configuration Files drzewa projektu).

j) W oknie edytora wywołaj menu kontekstowe i wybierz opcję Java Server Faces→Add Managed Bean.

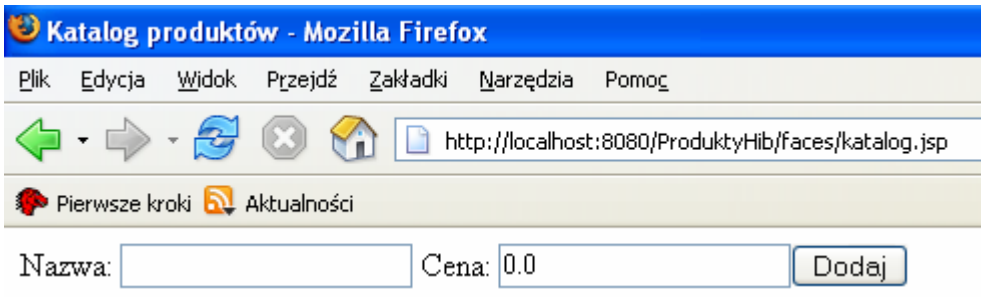


Jako nazwę komponentu podaj „katalogBean”, jako nazwę klasy „katalog.Katalog”, a jako zasięg „request”. Naciśnij przycisk Add.



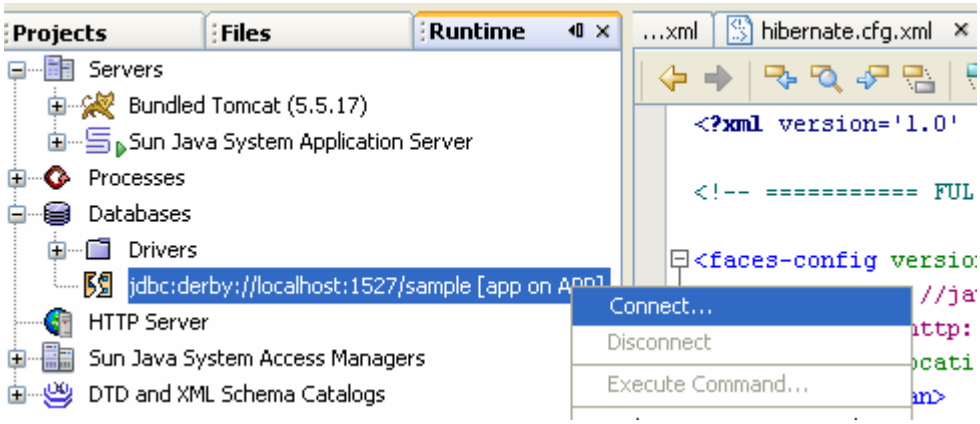
k) Zapisz wszystkie zmiany (File→Save All lub ikona w pasku narzędzi).

l) Uruchom stronę JSF wybierając z menu kontekstowego dla pliku `katalog.jsp` opcję Run File.

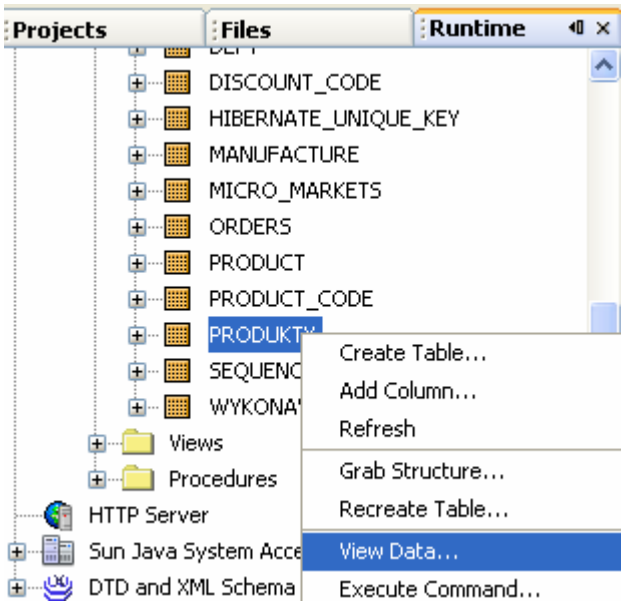


m) Dodaj za pomocą aplikacji dwa produkty do bazy danych.

n) Sprawdź czy wprowadzone dane produktów zostały zapisane do tabeli w bazie danych. W tym celu połącz się z bazą danych z poziomu środowiska NetBeans, wybierając opcję Connect z menu kontekstowego dla węzła reprezentującego wbudowany serwer bazy danych w panelu Runtime.



Następnie rozwiń gałąź Tables i dla tabeli PRODUKTY wybierz z menu kontekstowego opcję View Data.



8. Dodanie w klasie komponentu backing bean metody do odczytu listy produktów.

a) Dodaj w klasie Katalog dyrektywę importującą klasę java.util.List:

```
import java.util.List;
```

b) Dodaj w klasie Katalog metodę do odczytu z bazy danych za pomocą zapytania w języku HQL listy produktów:

```
    public List<Produkt> getProdukty() {
        SessionFactory sf = HibernateUtil.getSessionFactory();
        Session s = sf.openSession();
        List<Produkt> pp =
                    (List<Produkt>)s.createQuery("from Produkt").list();
        s.close();
        return pp;
    }
```

Nazwa metody getProdukty() została dobrana celowo, tak aby metoda ta udostępniała listę produktów jako właściwość produkty komponentu.

9. Dodanie na stronie JSF komponentu Data Table wyświetlającego tabelkę z produktami udostępnionymi przez komponent backing bean. Komponent Data Table będzie umieszczony w formularzu ponieważ w kolejnym punkcie ćwiczenia zostaną dodane do niego elementy, które wymagają otaczającego formularza.

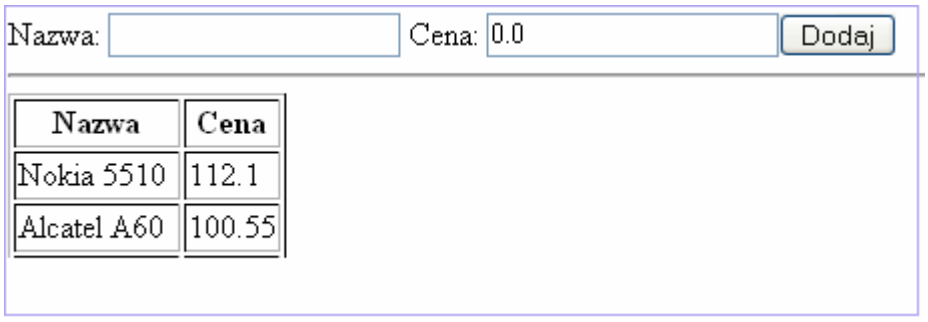
a) Dodaj na końcu kodu formularza (przed znacznikiem </h:form>) poniższy kod:

```
<HR>
<h:dataTable value="#{katalogBean.produkty}" var="prod" border="1">
  <h:column>
    <f:facet name="header">
      <h:outputText value="Nazwa"/>
    </f:facet>
    <h:outputText value="#{prod.nazwa}"/>
  </h:column>
  <h:column>
    <f:facet name="header">
      <f:verbatim>Cena</f:verbatim>
    </f:facet>
    <h:outputText value="#{prod.cena}"/>
  </h:column>
</h:dataTable>
```

b) Zapisz wszystkie zmiany (File→Save All lub ikona w pasku narzędzi).

c) Skompiluj projekt wybierając z menu kontekstowego dla projektu opcję Build Project.

d) Uruchom stronę JSF wybierając z menu kontekstowego dla pliku `katalog.jsp` opcję Run File.



10. Dodanie na stronie JSF w komponencie Data Table trzeciej kolumny zawierającej linki (obiekty Command Link) do usuwania poszczególnych produktów.

a) Dodaj przed znacznikiem </h:dataTable>) poniższy kod:

```
<h:column>
   <f:facet name="header">
     <f:verbatim></f:verbatim>
   </f:facet>
   <h:commandLink action="#{katalogBean.usun}">
     <h:outputText value="Usuń" />
     <f:param name="pid" value="#{prod.id}" />
   </h:commandLink>
 </h:column>
```

Zagnieżdżony w elemencie <h:commandLink> element <f:param> spowoduje przekazanie w żądaniu wygenerowanym przez kliknięcie linku parametru zawierającego identyfikator produktu do usunięcia. Metoda usun() komponentu backing bean, wskazana jako metoda obsługi zdarzenia kliknięcia linku, zostanie zaimplementowana w kolejnym kroku ćwiczenia.

11. Dodanie w klasie komponentu backing bean metody obsługującej zdarzenie usuwania produktu.

a) Dodaj w klasie Katalog metodę pomocniczą do usuwania z bazy danych produktu o podanym identyfikatorze:

```
public void deleteProdukt(Long pid) {
    SessionFactory sf = HibernateUtil.getSessionFactory();
    Session s = sf.openSession();
    Transaction tx = s.beginTransaction();
    Produkt p = (Produkt) s.get(Produkt.class, pid);
    s.delete(p);
    tx.commit();
    s.close();
}
```

b) Dodaj w klasie Katalog dyrektywę importującą klasę javax.faces.context.FacesContext:

```
import javax.faces.context.FacesContext;
```

c) Dodaj w klasie Katalog metodę do obsługi zdarzenia wybrania jednego z linków do usuwania produktu:

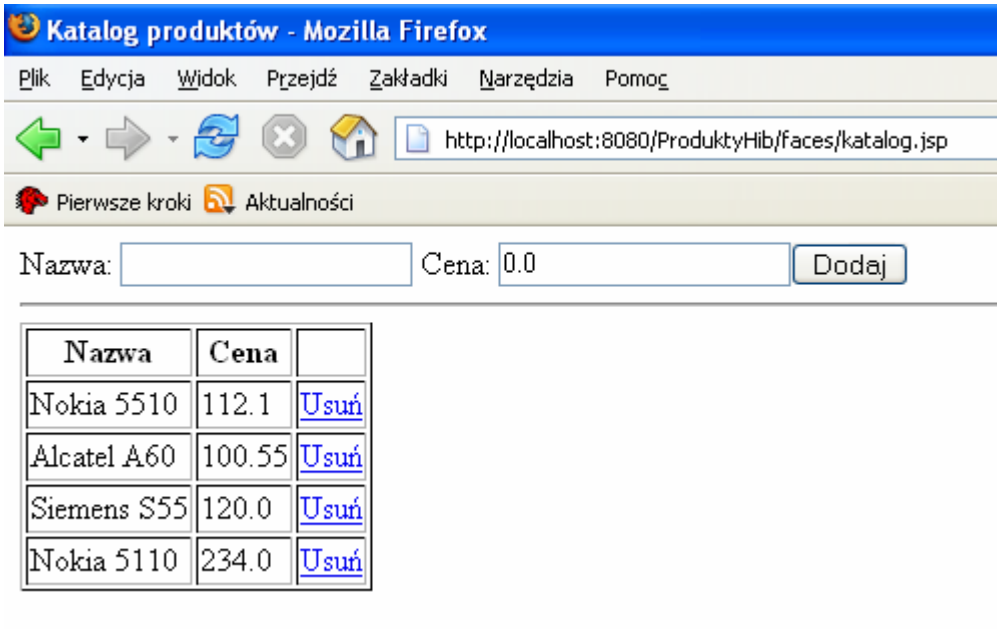
```
public String usun()
{
    FacesContext context = FacesContext.getCurrentInstance();
    String spid =
     context.getExternalContext().getRequestParameterMap().get("pid");
    Long pid = Long.parseLong(spid);
    deleteProdukt(pid);
    return null;
}
```

Metoda najpierw odczytuje poprzez FacesContext identyfikator produktu do usunięcia przekazany jako parametr w żądaniu, a następnie wywołuje metodę pomocniczą usuwającą produkt z bazy danych. Metoda zwraca null, tak aby po obsłużeniu zdarzenia ponownie została zaprezentowana ta sama strona aplikacji.

d) Zapisz wszystkie zmiany (File→Save All lub ikona w pasku narzędzi).

e) Skompiluj projekt wybierając z menu kontekstowego dla projektu opcję Build Project.

f) Uruchom stronę JSF wybierając z menu kontekstowego dla pliku `katalog.jsp` opcję Run File.



g) Przetestuj aplikację dodając i usuwając produkty.