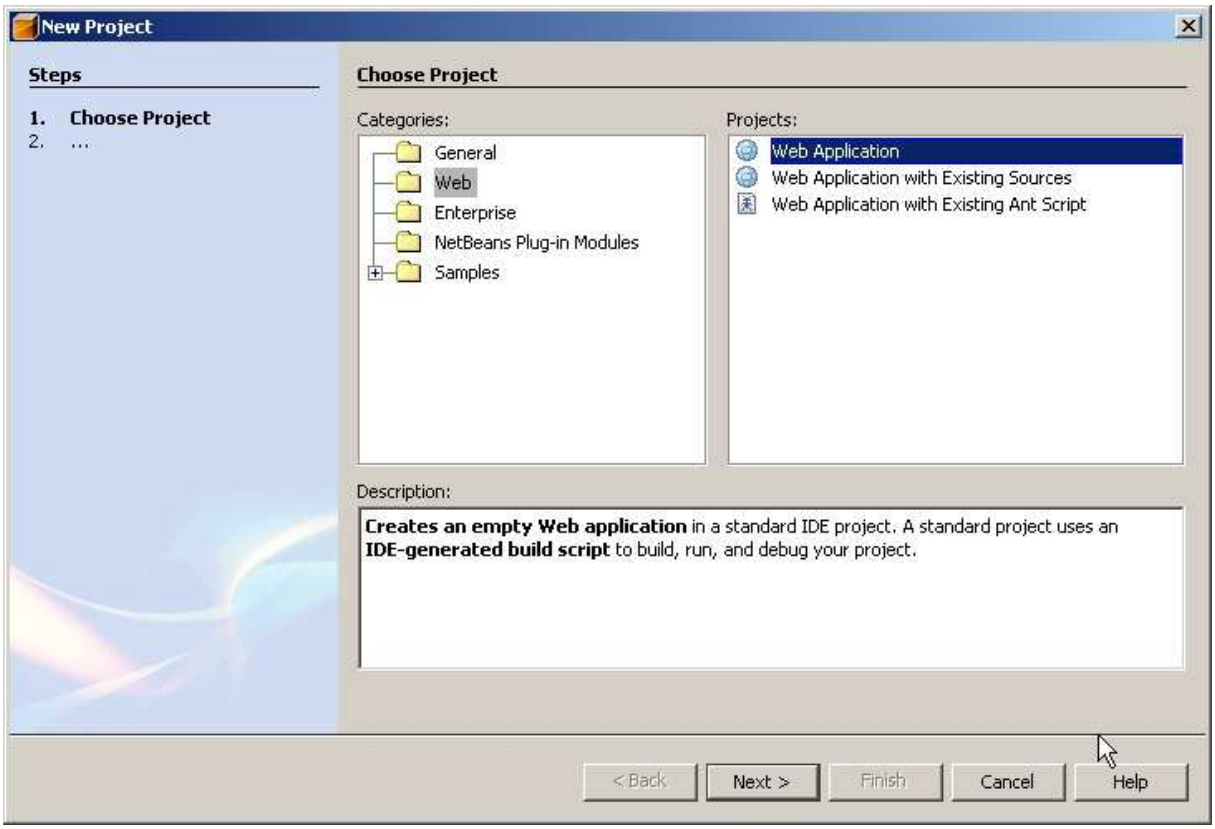
Aplikacje WWW - laboratorium

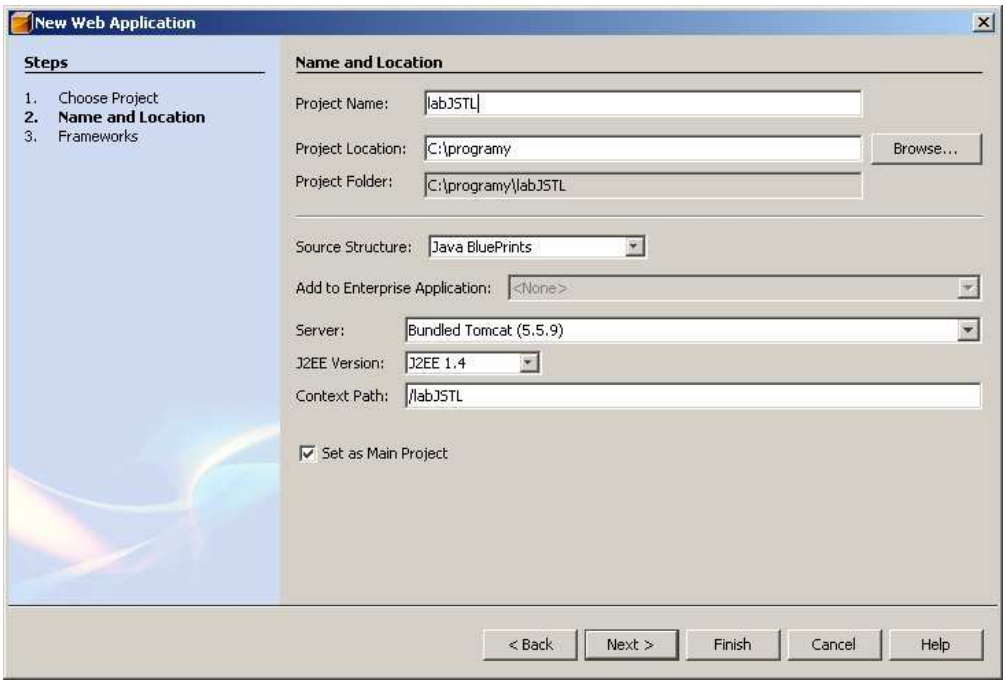
JavaServer Pages Standard Tag Library

Celem ćwiczenia jest zapoznanie ze standardową biblioteką znaczników JSTL. W ramach ćwiczenia zostanie skonstruowany prosty sklep internetowy przy użyciu stron JSP i komponentu JavaBean. Biblioteka JSTL zostania wykorzystana do przetwarzania pliku XML, sterowania przepływem pracy oraz wyświetlania danych i komunikacji z bazą danych. Do wykonania ćwiczenia potrzebne jest zintegrowane środowisko programistyczne NetBeans IDE (do pobrania z http://www.netbeans.org) oraz środowisko J2SE 1.5 (lub wyższe).

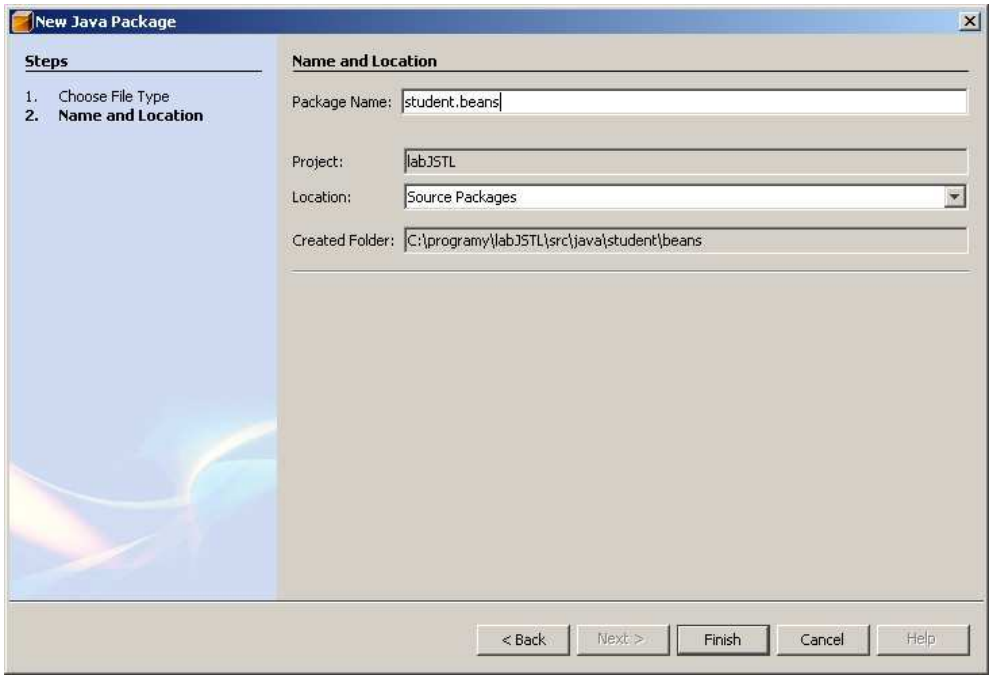
1. Uruchom narzędzie NetBeans IDE
2. Z menu głównego wybierz File→New Project. W oknie kreatora projektu wybierz kategorię Web i typ projektu Web Application. Kliknij przycisk Next >.



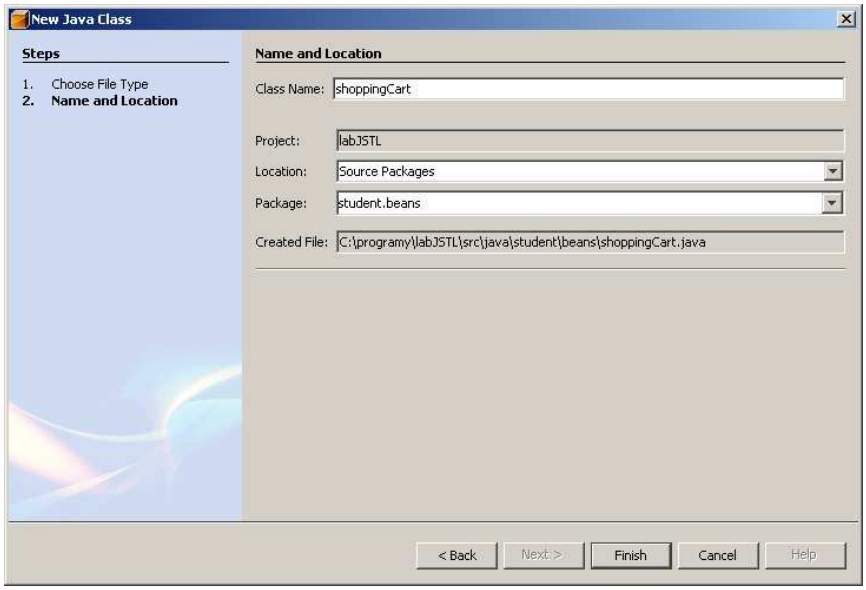
3. Podaj nazwę projektu, np. „labJSTL”. Kliknij przycisk Finish.



4. Budowany sklep internetowy będzie oferował funkcję koszyka, w którym klient może umieszczać (lub z niego usuwać) produkty. Ze strony osoby programującej wymaga to zapewnienia, że informacje o produktach wybranych przez klienta nie będą ulotne. W tym celu można zastosować komponent JavaBean (o zasięgu sesji) reprezentujący koszyk zakupów.
5. W nawigatorze projektu prawym przyciskiem myszy kliknij na węzeł Source Packages i wybierz opcję New→Java Package. Wprowadź nazwę pakietu, np. `student.beans`. Kliknij przycisk Finish.



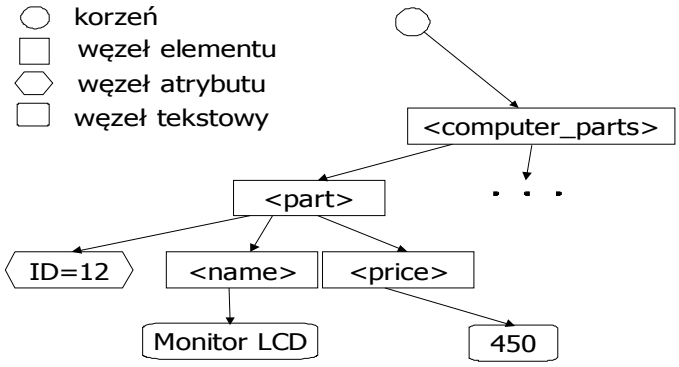
6. Pod węzłem Source Packages został utworzony węzeł student.beans. Kliknij na nim prawym przyciskiem myszy i wybierz New→Java Class. Wprowadź nazwę klasy, np. `shoppingCart` i kliknij przycisk Finish.



7. Przejdź do edycji utworzonej klasy (dwukrotne kliknięcie na nazwie klasy w nawigatorze projektu). Klasa będzie oferowała możliwość przechowywania identyfikatorów produktów w koszyku wraz z liczbą sztuk danego produktu. Takie dane można w łatwy sposób składować w tablicy haszowej, której kluczem jest identyfikator produktu, a wartością liczba zamówionych sztuk tego produktu. Ponadto, dostępne są metody do dodawania produktu (można dodawać po jednej sztuce), usuwania produktu (niezależnie od liczby sztuk) oraz pobierania liczby różnych produktów i całej zawartości koszyka. Zastąp zawartość klasy `shoppingCart` poniższym kodem.

```
package student.beans;

import java.util.ArrayList;

public class shoppingCart {
    private ArrayList items;

    public shoppingCart() { this.items = new ArrayList(); }

    public void add(int partID) { items.add(partID); }

    public void remove(int partID) {
        int position = items.indexOf(partID);
        items.remove(position);
    }

    public ArrayList getItems() { return this.items; }

    public int getSize() { return items.size(); }

}
```

8. Dane na temat towarów oferowanych w sklepie będą znajdowały się w pliku XML. Plik zawiera opis części komputerowych. Struktura drzewa DOM (z przykładowymi wartościami) tego dokumentu została przedstawiona poniżej.



9. Kliknij prawym przyciskiem na węzeł Web Pages i wybierz New→XML Document. Podaj nazwę pliku, np. `parts` (bez rozszerzenia). Kliknij przycisk Finish >. Przejdź do edycji pliku XML i wprowadź do niego poniższą zawartość.

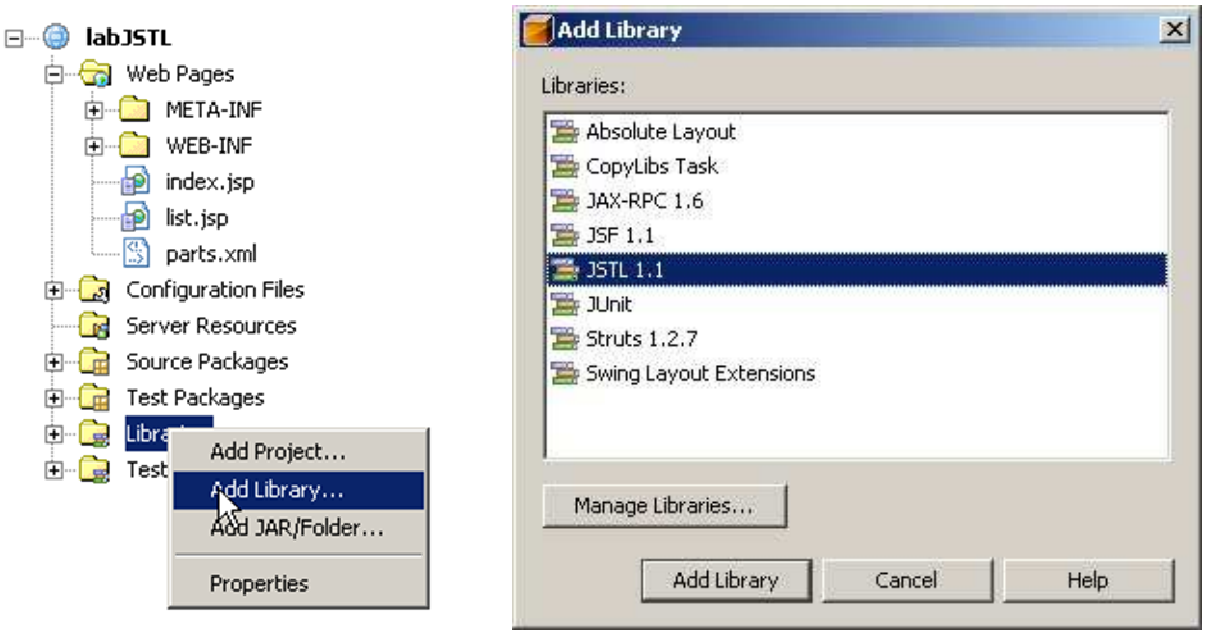
```
<?xml version="1.0" encoding="UTF-8"?>
<parts>
  <part ID="10">
    <name>Procesor 3.8 GHz</name>
    <price>450.50</price>
  </part>
  <part ID="25">
    <name>Mysz laserowa</name>
    <price>65.20</price>
  </part>
  <part ID="40">
    <name>Klawiatura multimedialna</name>
    <price>12</price>
  </part>
  <part ID="50">
    <name>Monitor LCD 19</name>
    <price>960</price>
  </part>
  <part ID="60">
    <name>Monitor CRT 17</name>
    <price>360</price>
  </part>
</parts>
```

10. Pierwszy przykład obrazuje sposób parsowania dokumentu XML za pomocą znaczników JSTL. Utwórz nowy dokument JSP i nazwij go `list.jsp`. Umieść w nim poniższy kod. Zwróć uwagę na sposób iteracji po kolekcji obiektów i użycie języka XPath.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
  <body>

  <h1>List of parts</h1>

  <c:import url="parts.xml" var="xmlFile"/>
  <x:parse doc="${xmlFile}" var="parsedFile"/>

  <ol>
    <x:forEach select="$parsedFile/parts/part" var="part">
      <li><b><x:out select="name"/></b>: <x:out select="price"/> </li>
    </x:forEach>
  </ol>

  </body>
</html>
```

11. W środowisku NetBeans zostanie wyświetlona informacja o wykrytym problemie. Użyte znaczniki nie mogą zostać rozpoznane. Jest to spowodowane tym, że biblioteka JSTL nie jest domyślnie włączona do projektu.

12. W nawigatorze projektu kliknij prawym przyciskiem na węzeł Libraries i wybierz Add Library. W okienku wyboru biblioteki wybierz JSTL 1.1 i kliknij przycisk Add Library. Dodaj też koniecznie bibliotekę Xalan!

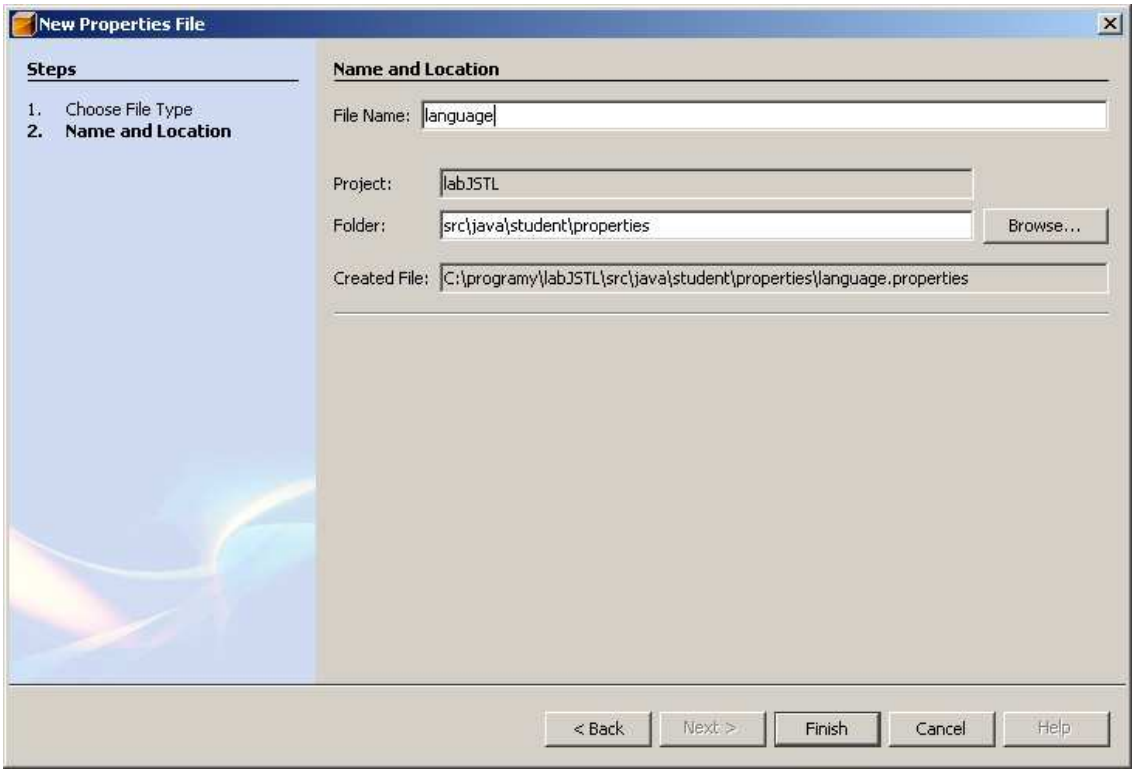


13. Uruchom i przetestuj stronę `list.jsp`.
14. Kolejnym krokiem jest utworzenie stron JSP, które będą służyły do zarządzania zawartością koszyka. Utworzony projekt automatycznie posiada główną stronę w pliku `index.jsp`. W nawigatorze projektu rozwiń węzeł Web Pages i przejdź do edycji głównej strony (dwukrotne kliknięcie myszą na nazwie pliku).

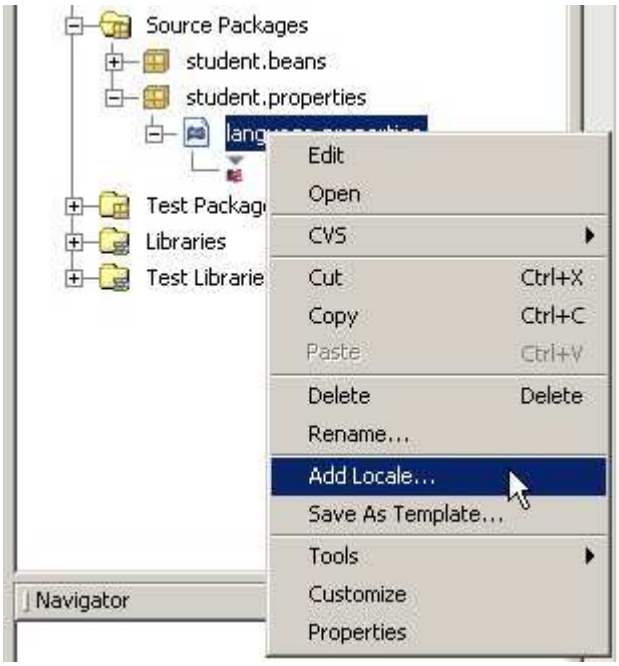
15. Na głównej stronie sklepu internetowego będą wyświetlone nazwy i ceny produktów, które zapisane są w pliku XML. Poniżej znajduje się kod głównej strony. Wszystkie potencjalne skryptlety zostały zastąpione odpowiednimi znacznikami JSTL. Umieść poniższy kod w pliku index.jsp.

```
<%@page contentType="text/html" pageEncoding="UTF-8" %>
<%@page import="java.util.Iterator"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
  <body>
  <jsp:useBean id="cart" class="student.beans.shoppingCart"
                         scope="session"/>
  <h1>Available parts</h1>

  <c:import var="xmlfile" url="parts.xml"/>
  <x:parse doc="${xmlfile}" var="doc"/>

  <table border=1>
  <tr><th>Name</th><th>Price</th><th> </th></tr>

  <x:forEach select="$doc/parts/part" var="part">
  <tr>
    <td><x:out select="$part/name"/></td>
    <td><x:out select="$part/price"/></td>

    <c:set var="part">
      <x:out select="$part/@ID"/>
    </c:set>
    <c:url value="/change_cart.jsp" var="addURL">
      <c:param name="action" value="add"/>
      <c:param name="partID" value="${part}"/>
    </c:url>

    <td><a href="${addURL}">add to cart</a></td>
  </tr>
  </x:forEach>
  </table>

  <br/>size of the cart: <c:out value="${cart.size}"/>
  <br/><a href="show_cart.jsp?action=show">show the cart</a>

  </body>
</html>
```

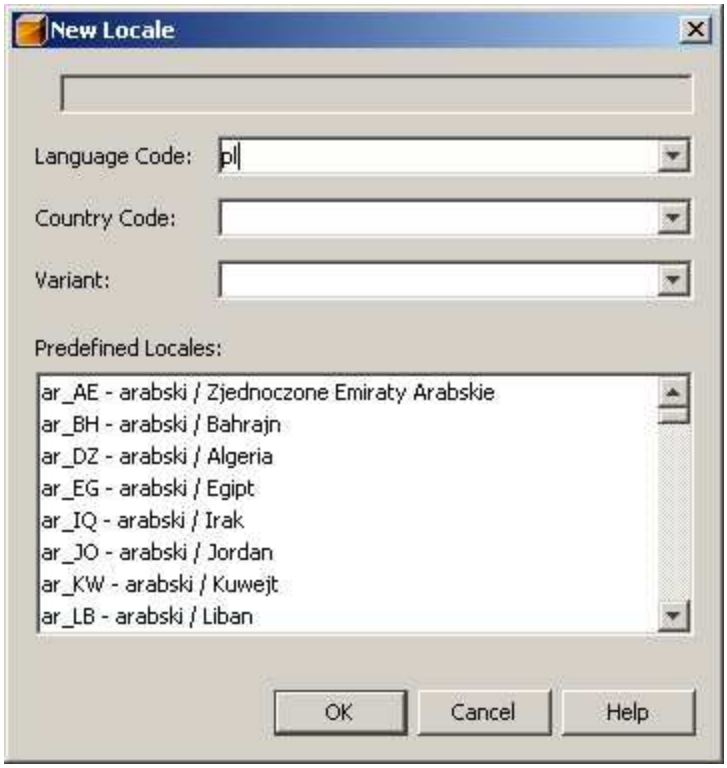
16. Następnym krokiem ćwiczenia jest dodanie odpowiednich wpisów dla różnych wersji językowych. W nawigatorze projektu dodaj pakiet `student.properties` (patrz opis dodania pakietu w punkcie 5). Kliknij prawym przyciskiem myszy na węźle utworzonego pakietu i wybierz New→Properties File. Wpisz nazwę pliku, np. `language` i kliknij przycisk Finish.



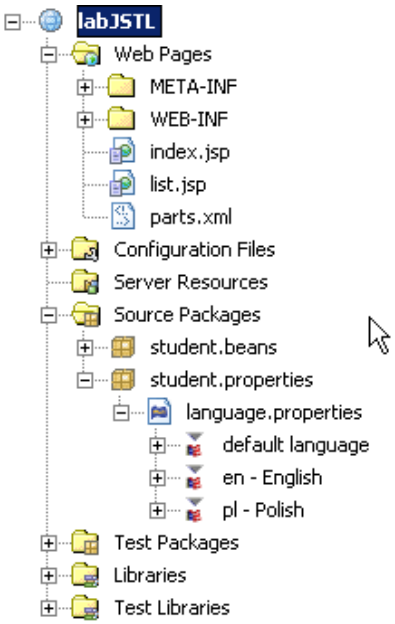
17. Rozwiń węzeł language.properties, kliknij na nim prawym przyciskiem myszy i wybierz Add Locale.



18. W polu kodu języka wpisz pl i kliknij przycisk OK.



19. Postępując tak samo, jak w kroku 19, dodaj lokację z kodem języka en. Ostatecznie w nawigatorze projektu powinna być następująca struktura.



20. Kliknij prawym przyciskiem myszy na węźle dla lokacji pl. Wybierz Add Property. W pole key wpisz nazwę klucza, przez który następuje odwołanie z dokumentu JSP do wartości (pole value) w odpowiednim pliku językowym, np. dla klucza „title” polska wartość to „Sklep”. Kliknij przycisk OK.

21. Postępując zgodnie z punktem 20 wprowadź do lokacji pl-polski następujące pary klucz-wartość.

```
title=Sklep
name=Nazwa
price=Cena
add_to_cart=Dodaj do koszyka
remove_from_cart=Usuń wszyskie
show_cart=Pokaż koszyk
added_to_cart=Dodano do koszyka.
num_of_items=Liczba różnych produktów w koszyku:
error=Błąd
cart=Zawartość koszyka
quantity=Ilość
go_back=Wróć
total_price=Wartość koszyka:
removed_from_cart=Usunięto z koszyka
```

22. Dla lokacji en-angielski i lokacji domyślnej wprowadź następujący tekst (możesz bezpośrednio skopiować i wkleić tekst do pliku z lokacją).

```
title=Shop
name=Name
price=Price
add_to_cart=Add to cart
remove_from_cart=Remove all
show_cart=Show cart
added_to_cart=Added to cart.
num_of_items=Number of different products in cart:
error=Error
removed_from_cart=Removed form cart.
cart=Cart
quantity=Quantity
go_back=Go back
total_price=Total price:
removed_from_cart=Removed from cart.
```

23. Pliki lokalizacji umożliwiają przygotowanie wielu wersji językowych wybieranych automatycznie przez serwer aplikacji na podstawie meta danych przesyłanych przez klienta HTTP. Powróć do pliku index.jsp i zamień wszystkie łańcuchy znaków na wywołania znaczników z biblioteki <fmt>.

```
<%@page contentType="text/html" pageEncoding="UTF-8" %>
<%@page import="java.util.Iterator"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
  <body>

  <jsp:useBean id="cart" class="student.beans.shoppingCart"
               scope="session"/>
  <fmt:setBundle basename="student.properties.language" />
  <h1><fmt:message key="title"/></h1>

  <c:import var="xmlfile" url="parts.xml"/>
  <x:parse doc="${xmlfile}" var="doc"/>

  <table border=1>
  <tr>
    <th><fmt:message key="name"/></th>
    <th><fmt:message key="price"/></th>
    <th> </th>
  </tr>

  <x:forEach select="$doc/parts/part" var="part">
  <tr>
    <td><x:out select="$part/name"/></td>
    <td><x:out select="$part/price"/></td>

    <c:set var="part">
      <x:out select="$part/@ID"/>
    </c:set>
    <c:url value="/ManageCart" var="addURL">
      <c:param name="action" value="add"/>
      <c:param name="partID" value="${part}"/>
    </c:url>

    <td><a href="${addURL}"><fmt:message key="add_to_cart"/></a></td>
  </tr>
  </x:forEach>
  </table>

  <br/><fmt:message key="num_of_items"/><c:out value="${cart.size}"/>
  <br/><a href="show_cart.jsp?action=show"><fmt:message
key="show_cart"/></a>

  </body>
</html>
```

24. Następnym krokiem jest zapewnienie możliwości dodawania i usuwania produktów z koszyka. Ponieważ jest to element logiki biznesowej, a nie prezentacji, implementujemy tę funkcjonalność w postaci serwletu. Kliknij węzeł korzenia projektu i wybierz New→servlet. Jako nazwę podaj `ManageCart` i umieść serwlet w pakiecie `student.servlets`. Kliknij przycisk Finish. Wypełnij serwlet poniższym kodem. Zwróć uwagę na przekazanie sterowania do strony JSP po wykonaniu logiki biznesowej. Wyświetl cechy projektu i w stronie Sources ustaw zgodność źródeł na 1.5 (dzięki czemu odbędzie się automatyczna konwersja między typami `int` i `Integer`).

```
package student.servlets;

import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

import student.beans.shoppingCart;

public class ManageCart extends HttpServlet {

  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
    throws ServletException, IOException {

    HttpSession session = request.getSession();
    shoppingCart cart = (shoppingCart)session.getAttribute("cart");

    String action = request.getParameter("action");
    int part = Integer.parseInt(request.getParameter("partID"));

    if (action.equals("add"))
      cart.add(part);
    else if (action.equals("remove"))
      cart.remove(part);

    response.sendRedirect("show_cart.jsp?action="+action);
  }
}
```

25. Ostatnim krokiem w budowie sklepu jest dodanie możliwości przeglądania produktów umieszczonych w koszyku klienta. Dodaj stronę JSP (patrz punkt 0) o nazwie `show_cart` i wprowadź do niej poniższy kod.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@taglib prefix="fmt" uri="http://java.sun.com/jstl/fmt"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x"%>

<html>
  <body>

  <jsp:useBean id="cart" class="student.beans.shoppingCart"
               scope="session"/>
   <fmt:setBundle basename="student.properties.language" />
   <c:import var="xmlfile" url="parts.xml"/>
   <x:parse doc="${xmlfile}" var="doc"/>

   <h3 style="color: red; align: center">
   <c:choose>
     <c:when test="${param.action =='add'}">
       <fmt:message key="added_to_cart"/>
     </c:when>
     <c:when test="${param.action == 'remove'}">
       <fmt:message key="removed_from_cart"/>
     </c:when>
     <c:when test="${param.action == 'show'}"/>
     <c:otherwise> <fmt:message key="error"/> </c:otherwise>
   </c:choose>
   </h3>

   <h2><fmt:message key="cart"/></h2>
   <table border="1">
     <tr>
       <th><fmt:message key="name"/></th>
       <th><fmt:message key="price"/></th>
       <th> </th>
     </tr>

   <c:forEach items="${cart.items}" var="item">
     <c:set var="id" value="${item}"/>

     <c:url value="/ManageCart" var="removeURL">
     <c:param name="action" value="remove"/>
     <c:param name="partID" value="${id}"/>
     </c:url>

     <tr>
       <td><x:out select="$doc//part[@ID=$id]/name"/></td>
       <td><x:out select="$doc//part[@ID=$id]/price"/></td>
       <td><a href="${removeURL}"><fmt:message
key="remove_from_cart"/></a></td>
     </tr>
   </c:forEach>
   </table>

   <a href="index.jsp"><fmt:message key="go_back"/></a>
</body>
</html>
```

26. Przejdź do edycji głównego pliku (`index.jsp`) i uruchom aplikację. Przetestuj działanie sklepu. Następnie, sprawdź działanie sklepu dla innej lokacji. Z głównego menu wybierz Narzędzia→Opcje internetowe, na zakładce Ogólne kliknij przycisk Języki. Umieść na pierwszym miejscu język angielski[en]. Uruchom stronę sklepu w przeglądarce.
27. Kolejny przykład stanowi prostą aplikację wykorzystującą znaczniki JSTL SQL i FMT do interakcji z bazą danych. Utwórz stronę JSP o nazwie `empSearch.jsp` i wypełnij ją poniższym kodem, a następnie uruchom i przetestuj działanie. Przed uruchomieniem dodaj do projektu sterowniki JDBC do bazy danych Oracle.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>

<html>
    <body>
    <h1>Search form</h1>
    <form action="empSearch.jsp">
        Name: <input type="text" name="empName"/>
        <input type="submit" value="search"/>
    </form>

    <sql:setDataSource url="jdbc:oracle:thin:@oracle-
server.edu.wsnhid.pl:1521:orcl"
      user="scott" password="tiger" driver="oracle.jdbc.OracleDriver"/>

    <c:choose>
      <c:when test="${!empty param.empName}">
        <sql:query var="employees">
        SELECT * FROM pracownicy WHERE nazwisko LIKE UPPER(?)
          <sql:param value="${param.empName}%"/>
        </sql:query>
      </c:when>
      <c:otherwise>
        <sql:query var="employees">
        SELECT * FROM pracownicy
        </sql:query>
      </c:otherwise>
    </c:choose>

    <table border="1">
    <tr>
      <th>NAZWISKO</th><th>ETAT</th><th>ZATRUDNIONY</th><th>PLACA</th>
    </tr>
    <c:forEach items="${employees.rows}" var="employee">
    <tr>
      <td><c:out value="${employee.nazwisko}"/></td>
      <td><c:out value="${employee.etat}"/></td>
      <td><fmt:formatDate dateStyle="long"
value="${employee.zatrudniony}"/></td>
      <td><fmt:formatNumber type="currency"
value="${employee.placa_pod}"/></td>
    </tr>
  </c:forEach>
  </table>
</body>
</html>
```

28. Ostatnie zadanie polega na przygotowaniu własnego znacznika, który będzie ułatwiał drukowanie informacji o książkach. Kliknij prawym klawiszem myszy na ikonie folderu `WEB-INF`. Z menu wybierz New→File/Folder→Other→Folder i dodaj folder o nazwie `tags`. Kliknij prawym klawiszem myszy na nowo utworzonym folderze i z menu wybierz New→File/Folder→Web→Tag File. Jako nazwę pliku podaj `book`. Kliknij przycisk Finish. Edytuj plik `book.tag` i umieść w nim poniższy kod

```
<%@tag pageEncoding="UTF-8"%>

<%@attribute name="title" required="true"%>
<%@attribute name="author"%>

<table border="1" width="500">
    <tr>
        <td align="center" colspan="2"><h3>${title}</h3></td>
    </tr>
    <tr>
        <td><em>Author:</em></td><td><strong>${author}</strong></td>
    </tr>
    <tr>
        <td><em>Book Info:</em></td><td><jsp:doBody/></td>
    </tr>
</table>
```

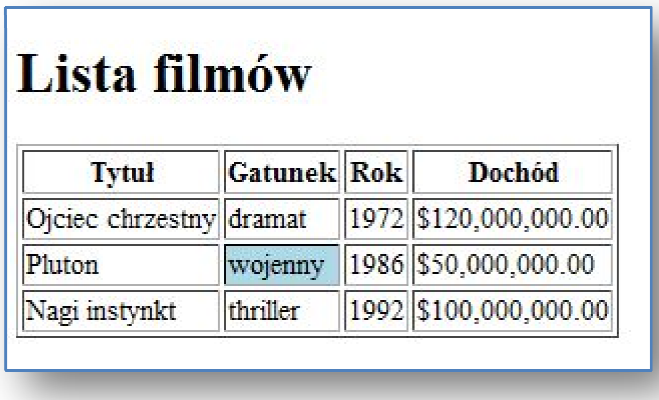
29. Utwórz nową stronę JSP o nazwie `customTag.jsp`. Umieść w niej poniższy kod. Uruchom stronę i zaobserwuj efekt działania.

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<%@taglib tagdir="/WEB-INF/tags" prefix="my"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <title>Custom tag example</title>
    </head>
    <body>

    <h1>Custom tag example</h1>

    <my:book title="Fundamentals of Database Systems"
             author="Elmasri, Navathe">
        This book combines clear explanations of theory and design,
        broad coverage of models and real systems, and excellent
        examples with up-to-date introductions to modern database
        technologies.
    </my:book>

    </body>
</html>
```

Zadanie do samodzielnego wykonania

Otwórz dostarczony przez prowadzącego szkielet projektu `JSTL_zadanie.zip` i rozpakuj projekt, a następnie otwórz projekt w środowisku NetBeans. Zapoznaj się z zawartością projektu. Zauważ, że serwlet pełniący rolę kontrolera przygotowuje listę filmów i umieszcza ją w postaci komponentu JavaBean o nazwie **`filmy`** w zasięgu widzialności `request`.

Przygotuj stronę JSP o nazwie strona.jsp, która odczyta zawartość listy filmów i wyświetli ją w postaci tabelarycznej. Posłuż się znacznikami `<c:forEach>`, `<c:out>`, `<c:choose>`, `<c:when>` i `<c:otherwise>` do przygotowania strony. Zwróć uwagę, że dochody uzyskiwane przez filmy są wyświetlone jak pieniądze, posłuż się znacznikiem `<fmt:formatNumber>` do poprawnego sformatowania tych wartości. Jeśli gatunek filmu to „wojenny", zmień tło komórki tabeli na inny kolor. Twoja aplikacja powinna wyglądać następująco:



Lista filmów

Tytuł	Gatunek	Rok	Dochód
Ojciec chrzestny	dramat	1972	$120,000,000.00
Pluton	wojenny	1986	$50,000,000.00
Nagi instynkt	thriller	1992	$100,000,000.00