

# Rozproszone systemy internetowe

## RESTful web services, JAX-RS

Zagadnienia:

1. Tworzenie RESTful serwisu, zwracanie listy obiektów
2. WADL
3. Tworzenie klienta (JAX-RS Client)
4. Przekazywanie obiektu jako parametr dla serwisu

Sources and help:

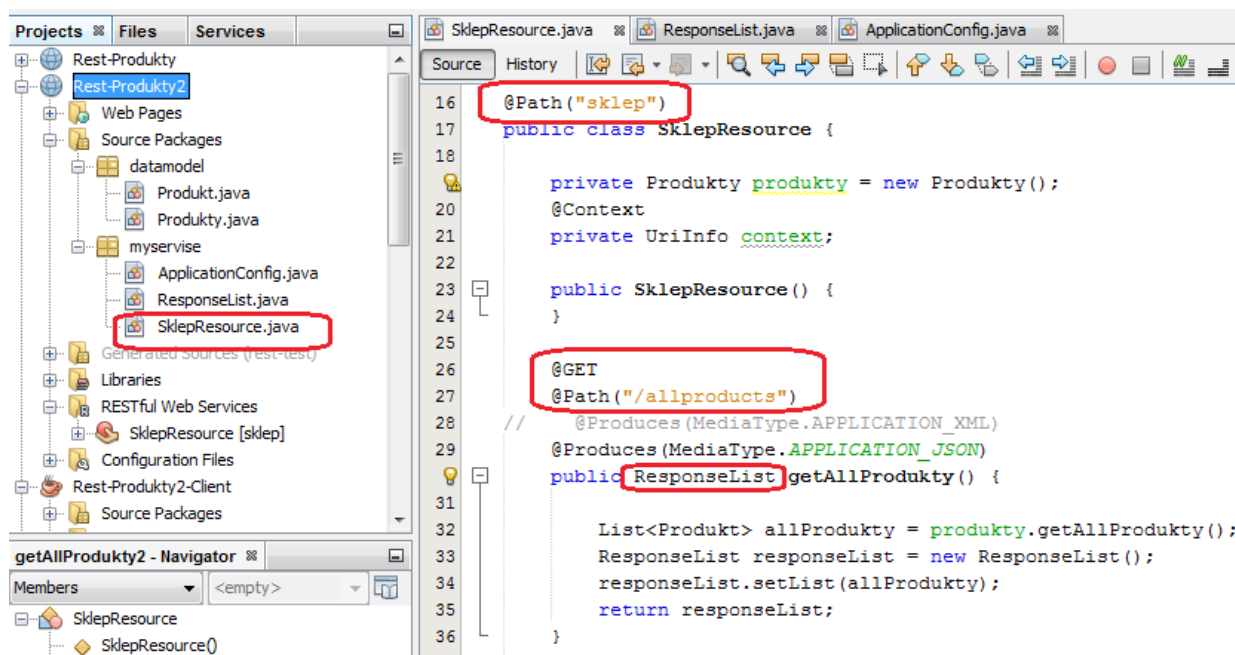
What are RESTful Web Services (ORACLE)

Jersey documentation - 5. Client API

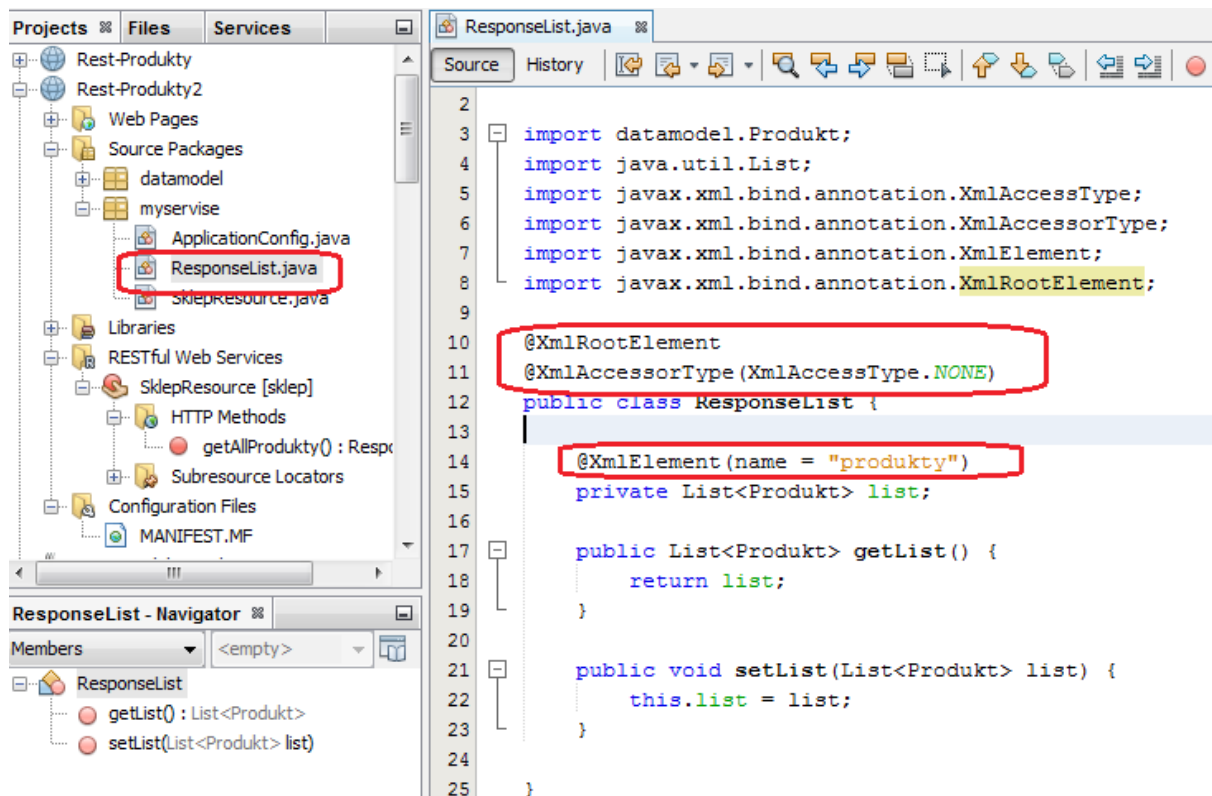
Jersey User Guide

## Ćwiczenie 1. Tworzenie serwisu zwracającego listę produktów w JSON

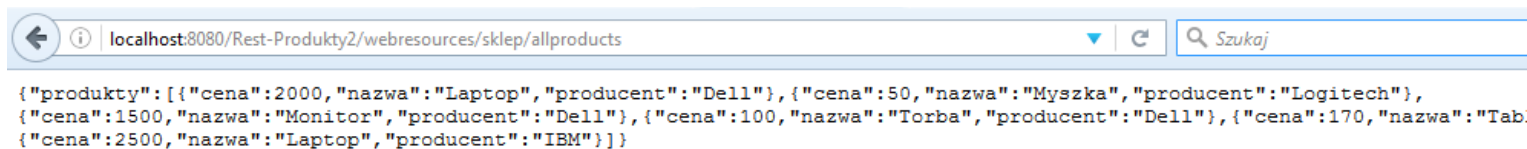
Utwórz RESTful serwis zwracający listę produktów



Obiekt odpowiedzi ResponseList

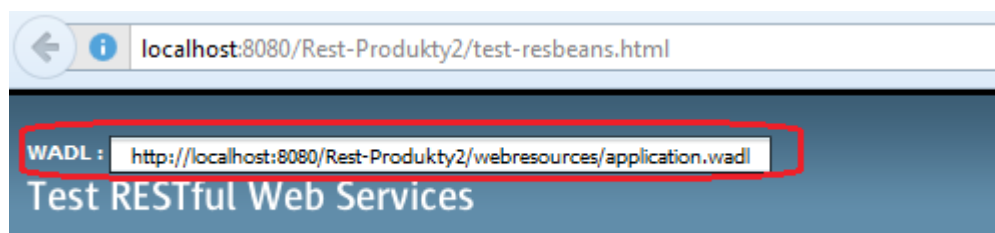
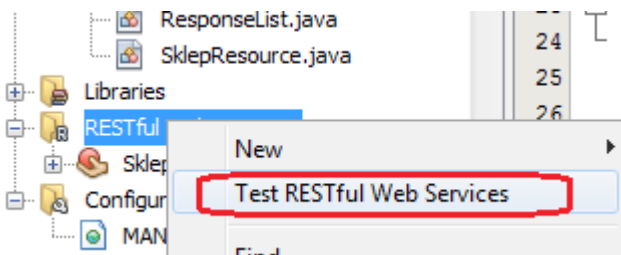


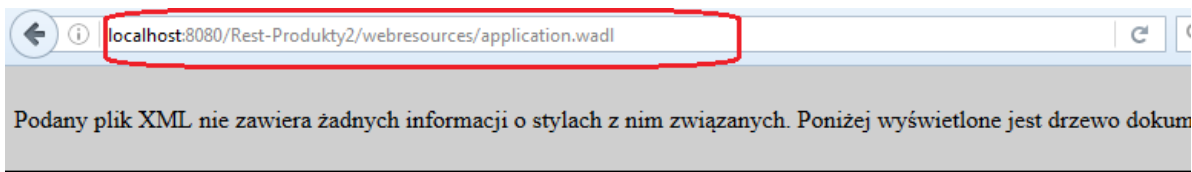
Testowanie serwisu w przeglądarce



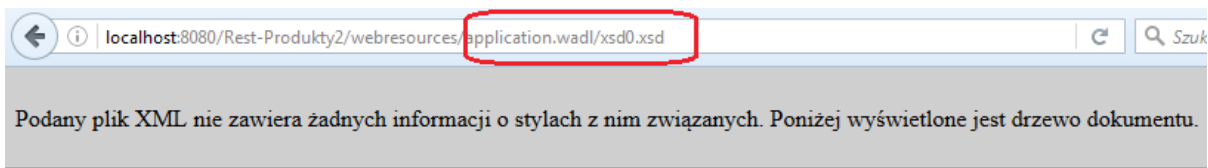
## Ćwiczenie 2. Podgląd pliku WADL dla serwisu

Przejrzyj zawartość pliku WADL





```
- <application>
  <doc jersey:generatedBy="Jersey: 2.10.4 2014-08-08 15:09:00"/>
  <doc jersey:hint="This is simplified WADL with user and core resources only. To get full WADL with extended re
  Produkty2/webresources/application.wadl?detail=true"/>
  - <grammars>
    - <include href="application.wadl/xsd0.xsd">
      <doc title="Generated" xml:lang="en"/>
    </include>
  </grammars>
  - <resources base="http://localhost:8080/Rest-Produkty2/webresources/">
    - <resource path="sklep">
      - <resource path="/allproducts">
        - <method id="getAllProdukty" name="GET">
          - <response>
            <ns2:representation element="responseList" mediaType="application/json"/>
          </response>
        </method>
      </resource>
    </resource>
  </resources>
XML Schema
```

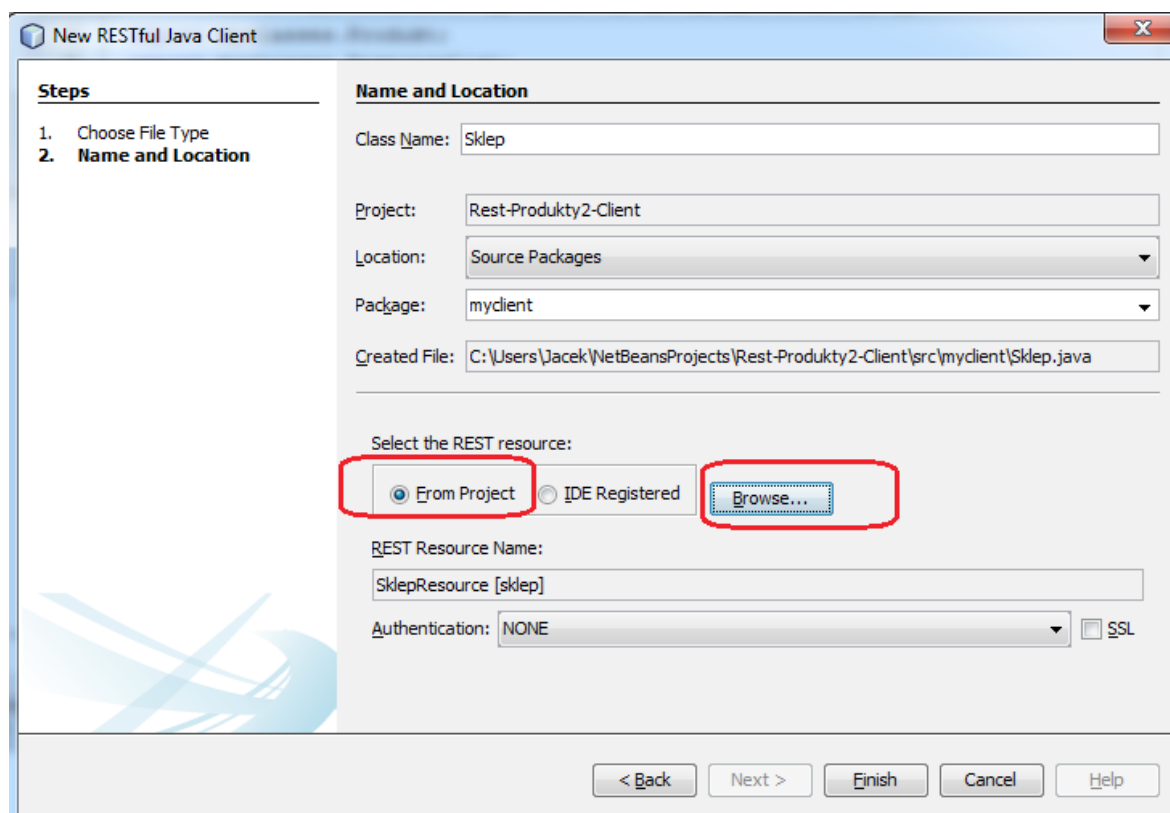


```
- <xs:schema version="1.0">
  <xs:element name="produkt" type="produkt"/>
  <xs:element name="responseList" type="responseList"/>
  - <xs:complexType name="responseList">
    - <xs:sequence>
      <xs:element name="list" type="produkt" nillable="true" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  - <xs:complexType name="produkt">
    - <xs:sequence>
      <xs:element name="cena" type="xs:int"/>
      <xs:element name="nazwa" type="xs:string" minOccurs="0"/>
      <xs:element name="producent" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

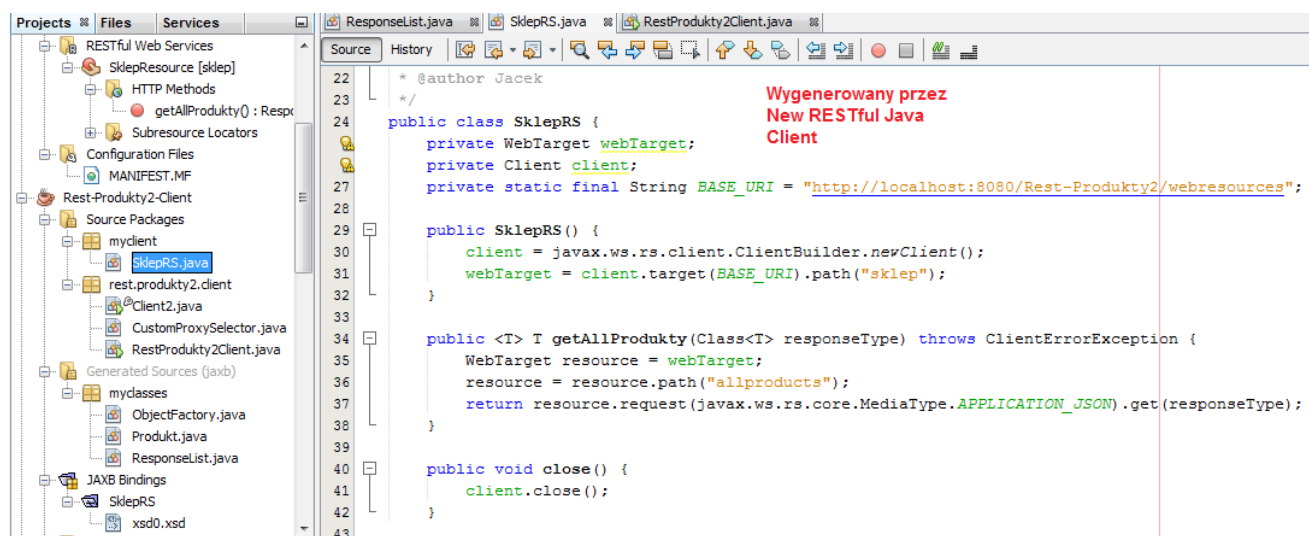
### Ćwiczenie 3. Tworzenie klienta dla serwisu (Jersey Client)

Utwórz klienta dla serwisu zwracającego produkty

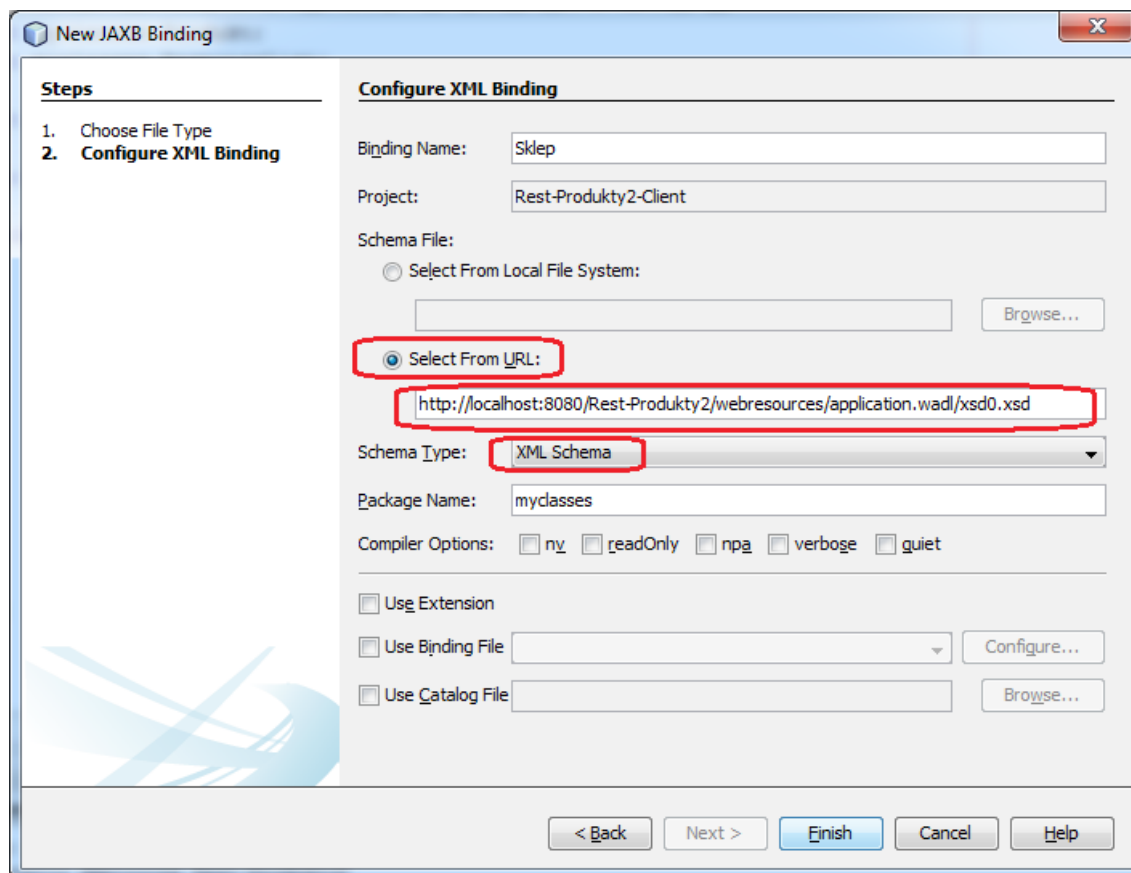
1. Tworzenie klienta „New RESTful Java Client” wizard



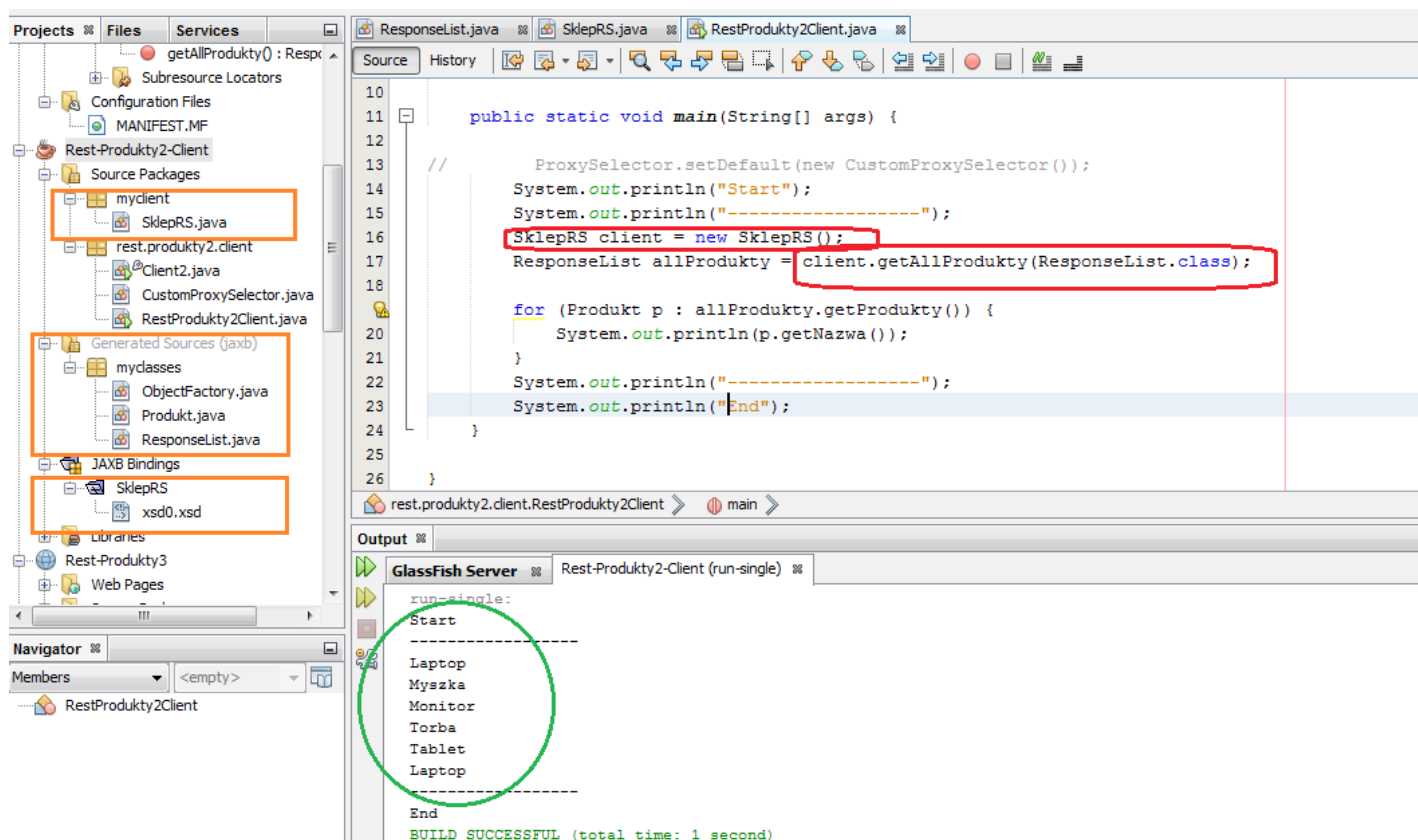
Przykładowa treść wygenerowanego klienta



2. Użycie JAXB Binding do utworzenia potrzebnych klas (np. ResponseList.java) w kliencie



3. Użycie klienta do pobrania listy produktów



Ktoś ze studentów zgłaszał, że był problem ze zwracaniem obiektem, był pusty (null), przy Glassfish 5.0. Przy Glassfish 4.1 działa. Rozwiązaniem się okazało ustawienie `@Produces(MediaType.APPLICATION_XML)` po stronie serwisu zamiast `@Produces(MediaType.APPLICATION_JSON)`.

Podgląd komunikatów w HTTP Monitorze:

HTTP Monitor [REST Project 1]

Traffic Log

Request Host: - all - Target Host: - all - Interface: - all - Operation: - all -

| Count. | Time              | Request Host | Target Host | Interface   | Operation   | Time Taken | Req Sz | 316 |
|--------|-------------------|--------------|-------------|-------------|-------------|------------|--------|-----|
| 0      | 2016-05-23 11:... | localhost    | localhost   | - unknown - | - unknown - | 16         | -1     |     |

GET http://localhost:8080/Rest-Produkty2/webresources/sklep/allproducts HTTP/1.1  
 Accept: application/json  
 User-Agent: Jersey/2.5.1 (HttpURLConnection 1.8.0\_66)  
 Host: localhost:8080  
 Proxy-Connection: keep-alive

```

1 {"produkty": [
2   {
3     "cena": 2000,
4     "nazwa": "Laptop",
5     "producent": "Dell"
6   },
7   {
8     "cena": 50,
9     "nazwa": "Myszka",
10    "producent": "Logitech"
11  },
12  {
13    "cena": 1500,
14    "nazwa": "Monitor",
15    "producent": "Dell"
16  },
17  {
18    "cena": 100,
19    "nazwa": "Torba",
  
```

## Ćwiczenie 4. Serwis z przyjmujący obiekt jako parametr i zwracający listę obiektów

Dodanie RESTful API wyszukujące produkty po nazwie, producencie i cenie

Projects | Files | Services

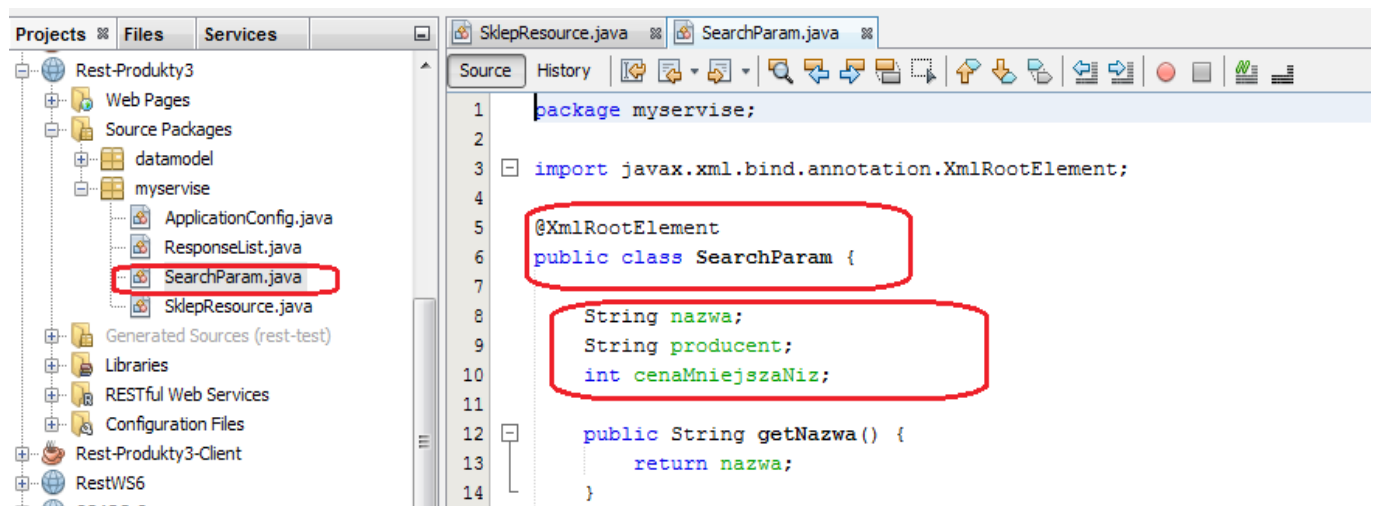
Rest-Produkty3

- Web Pages
- Source Packages
  - datamodel
  - myservice
    - ApplicationConfig.java
    - ResponseList.java
    - SearchParam.java
    - SklepResource.java
- Generated Sources (rest-test)
- Libraries
- RESTful Web Services
- Configuration Files
- Rest-Produkty3-Client
- RestWS6
- SOAPCz2
- testClassFilesToMapV04\_5
- WebServices

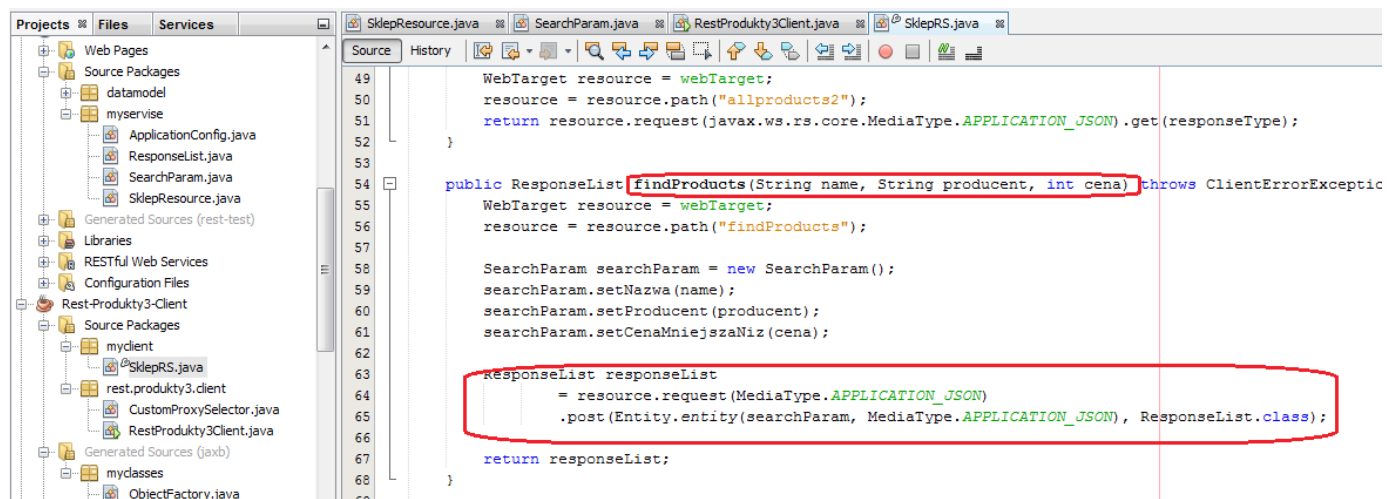
SklepResource.java

```

46     return responseList;
47   }
48 }
49
50 @POST
51 @Path("/findProducts")
52 @Produces(MediaType.APPLICATION_JSON)
53 @Consumes(MediaType.APPLICATION_JSON)
54 public ResponseList findProducts(SearchParam searchElement) {
55     ResponseList responseList = new ResponseList();
56     List<Produkt> resLista
57         = produkty.findProduktyByNazwaProducentCenaMniejszaNiz(searchElement.getNazwa(), searchElement.getProducent(), searchElement.getCena());
58     responseList.setList(resLista);
59
60     return responseList;
61 }
62 //
  
```



Zapytanie po stronie klienta



Podgląd komunikatów w HTTP Monitorze:

