



**AGH**

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Fizyki i Informatyki Stosowanej

---

# **Zaawansowane techniki internetowe**

**Nabiz**

kierunek studiów: **informatyka stosowana**

## **AspectJ - wdrożenie przykładów**

**Kraków, 04.05.2021**

<b>Wstęp</b>	<b>3</b>
<b>Przykład 1.</b>	<b>4</b>
<b>Przykład 2.</b>	<b>8</b>
<b>Przykład 3.</b>	<b>13</b>
<b>Posłowie</b>	<b>15</b>

# Wstęp

Dokument zawiera instrukcję wdrożenia przykładów prezentowanych na seminarium dotyczącym programowania aspektowego i języka AspectJ. Tekst ten nie zawiera omówienia przykładów, są to tylko informacje dotyczące wdrożenia przykładów.

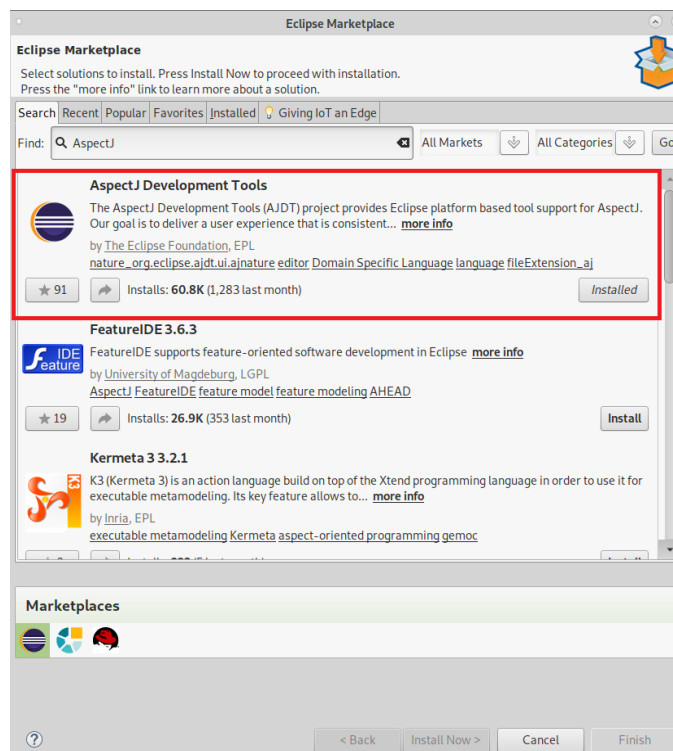
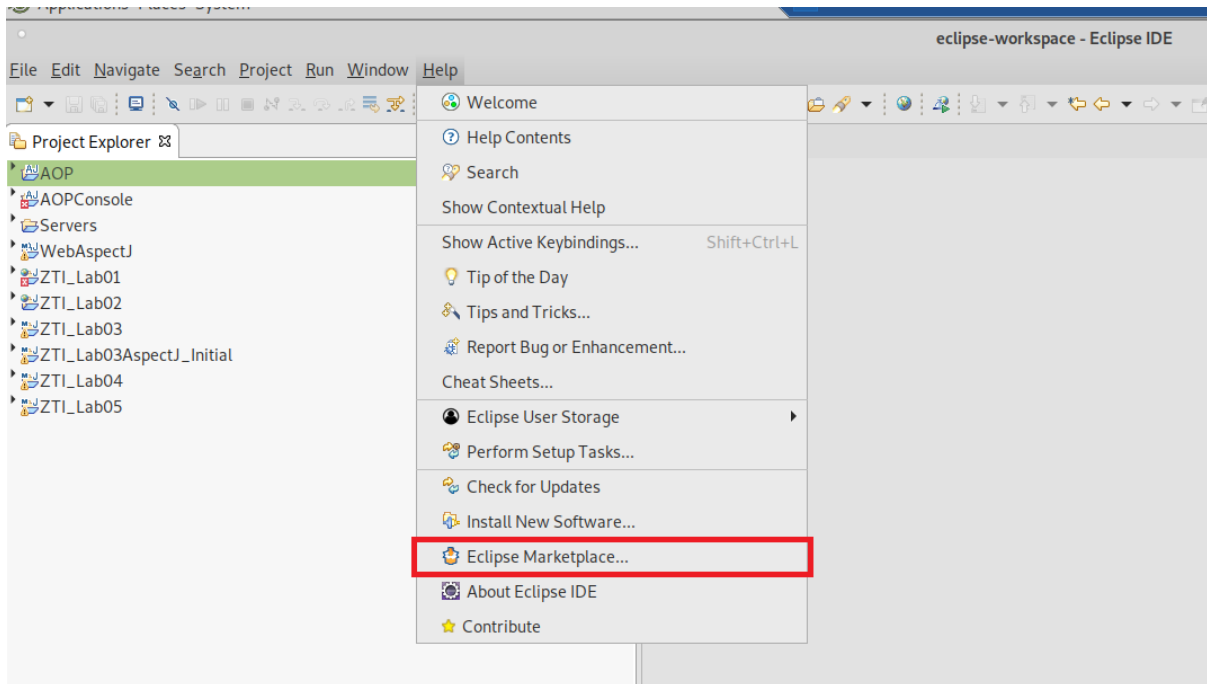
**Przykład 1** - prosta aplikacja konsolowa mająca na celu prezentację technologii AspectJ.

**Przykład 2** - aplikacja webowa oparta na servletach, w której wykorzystano aspekt do obsługi połączenia z bazą danych. W przykładzie obiekt do połączenia z bazą danych zapisujemy w polu odpowiedniej klasy bazowej.

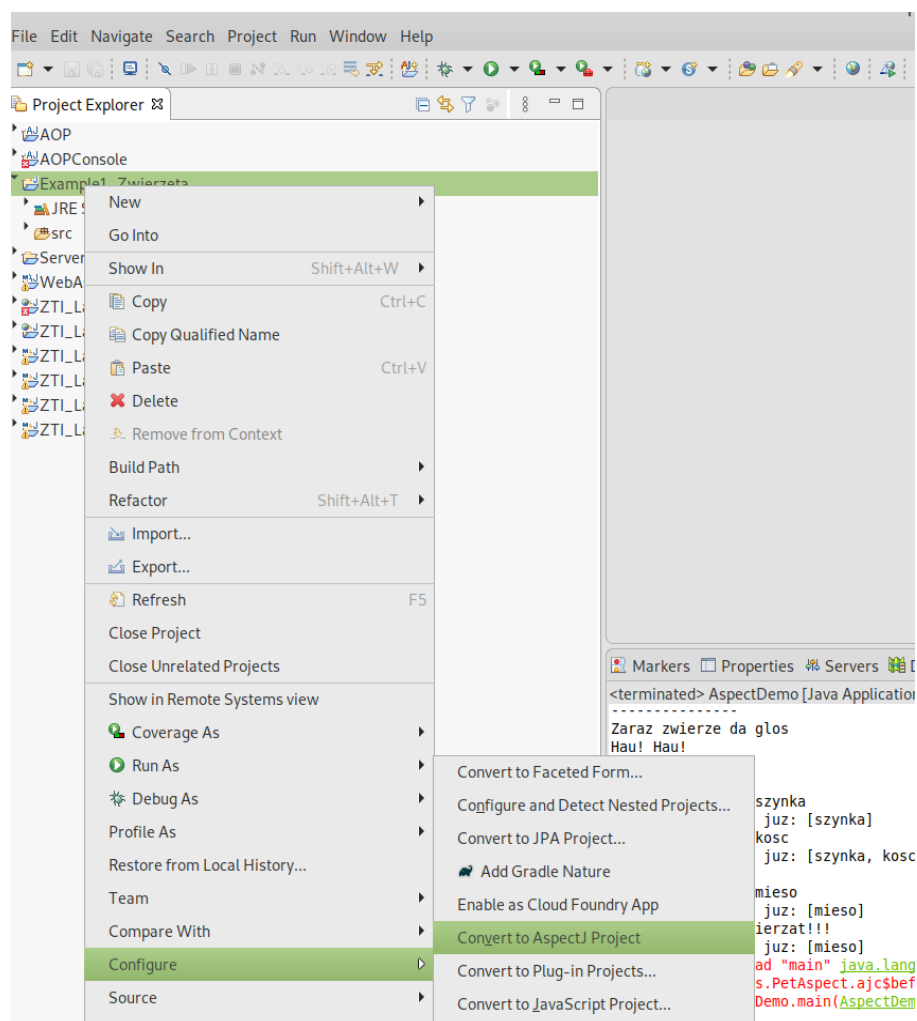
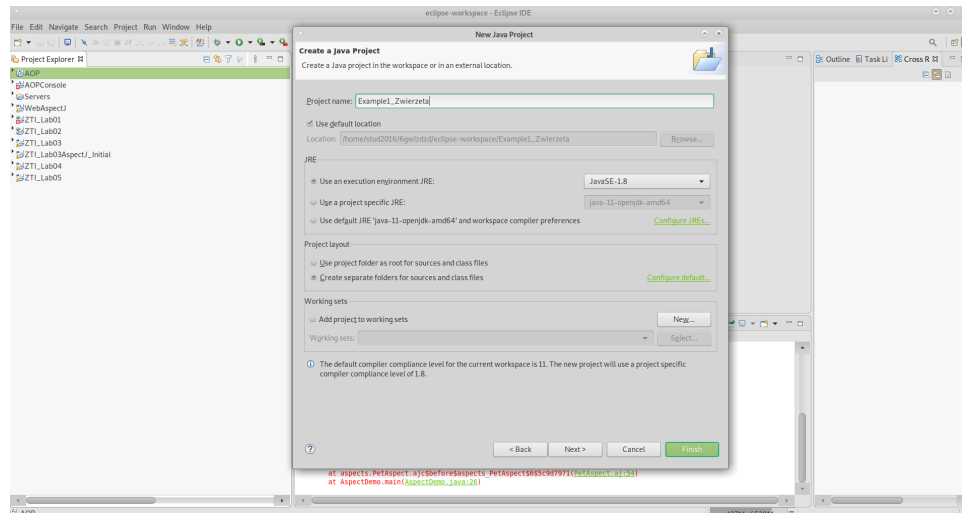
**Przykład 3** - ta sama aplikacja co z przykładu drugiego, ale wykonana inną metodą. W tym przykładzie obiekt do połączenia z bazą danych podajemy jako parametr wywołania odpowiedniej metody.

# Przykład 1.

Na początku musimy zainstalować dodatek AJDT w Eclipse, który pozwoli na tworzenie aspektów i umożliwi kompilację projektów.

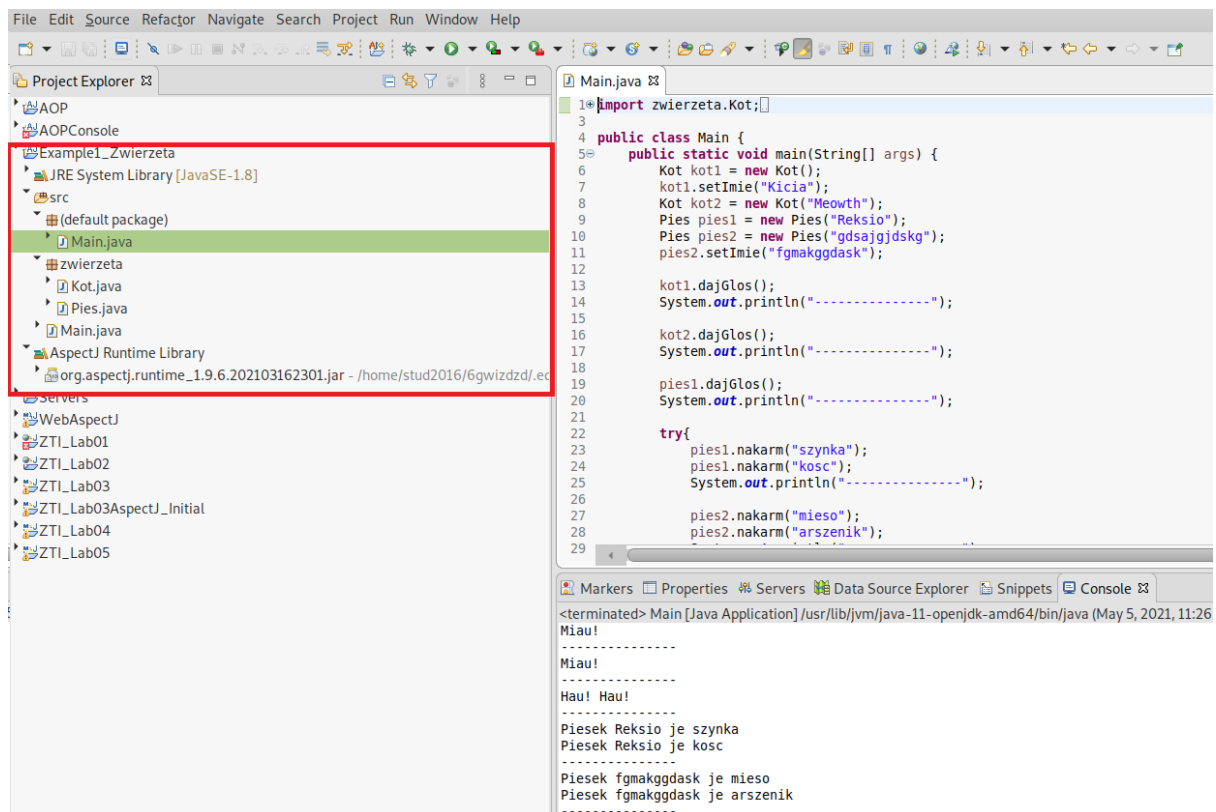


Tworzymy nowy Java Project, a następnie konwertujemy go do projektu AspectJ. Do projektu zostanie dodana odpowiednia biblioteka umożliwiającą uruchomienie projektu z aspektami.



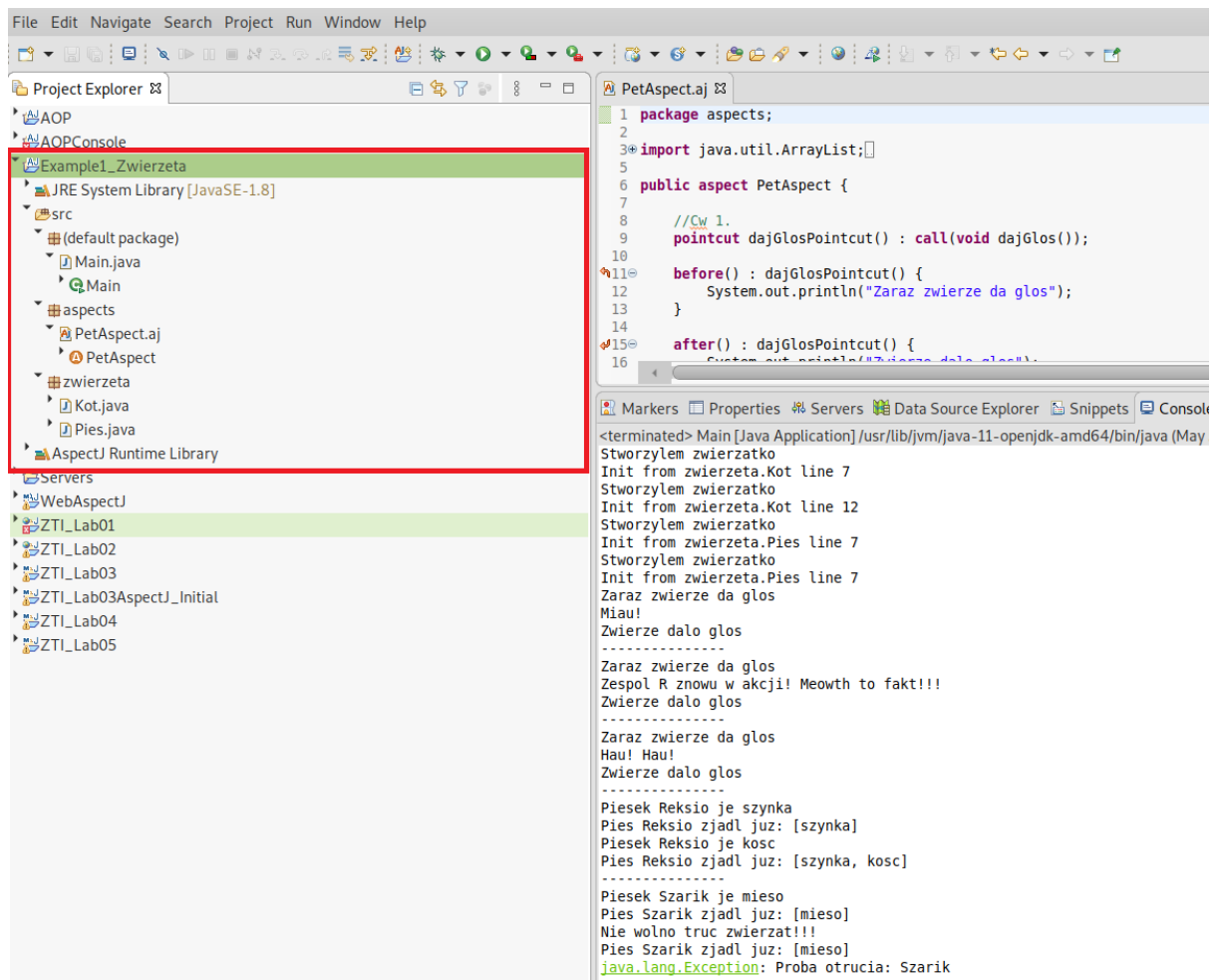
W katalogu **Example1\_Zwierzeta/src** znajdują się pliki źródłowe, które należy skopiować do katalogu **src** naszego projektu. Na początku tworzymy paczkę **zwierzeta** i kopiujemy do niej pliki **Kot.java** i **Pies.java**, następnie kopiujemy **Main.java** bezpośrednio do katalogu **src** (default package).

Po uruchomieniu aplikacji:



Następnie dodajemy paczkę **aspects** i plik **PetAspect.aj**.

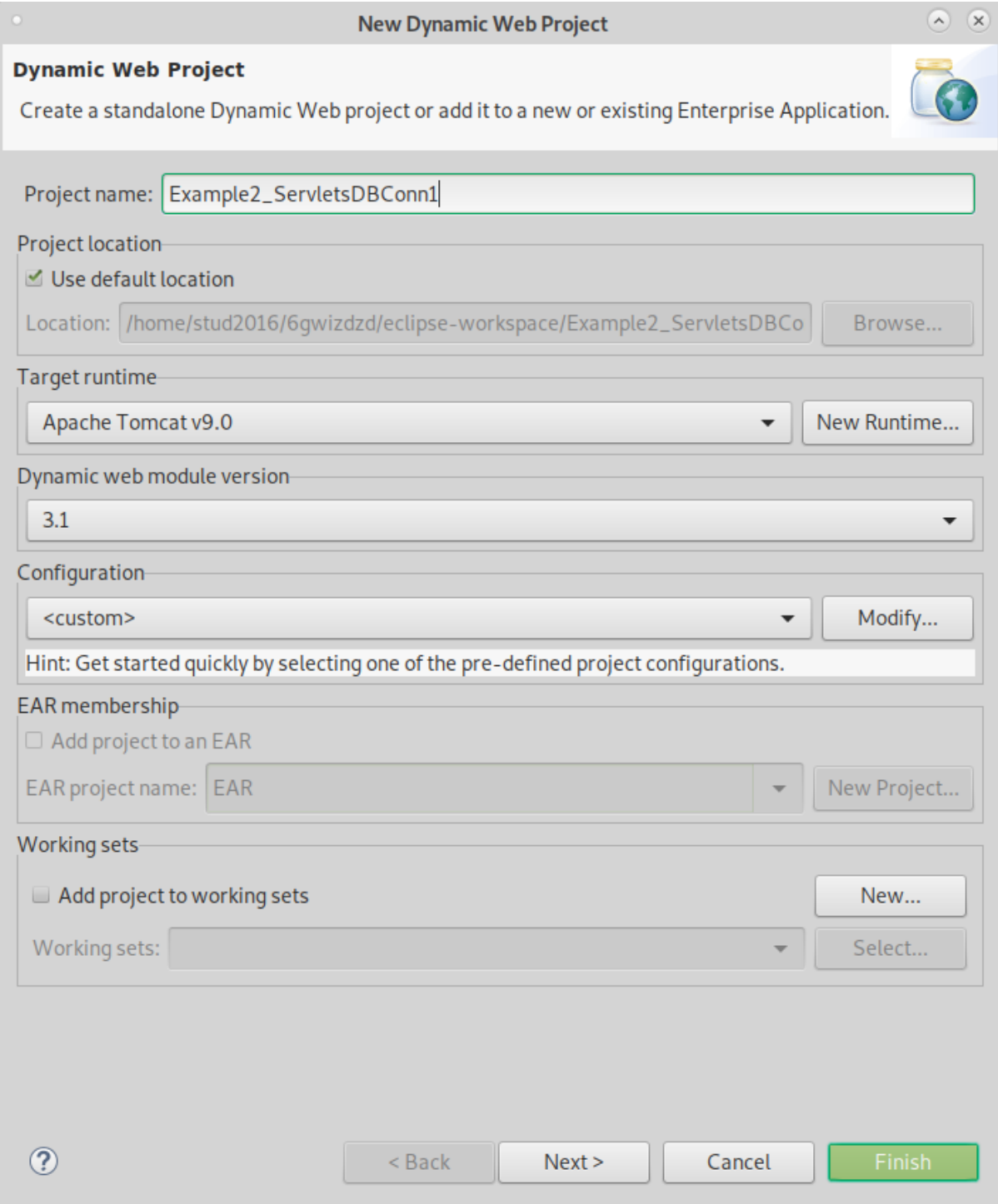
Dodanie aspektów zmienia działanie aplikacji:



## Przykład 2.

Kolejny przykład to wykorzystanie aspektów do obsługi połączenia z bazą danych dla dwóch prostych servletów wypisujących przykładowe dane.

Tworzymy nowy Dynamic Web Project. Następnie konwertujemy go do projektu AspectJ tak samo jak w pierwszym przykładzie!



The screenshot shows the 'New Dynamic Web Project' dialog box in Eclipse. The title bar reads 'New Dynamic Web Project'. The main heading is 'Dynamic Web Project' with a subtitle 'Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.' and a small icon of a jar and globe. The dialog is divided into several sections:

- Project name:** A text field containing 'Example2\_ServletsDBConn1'.
- Project location:** A section with a checked checkbox 'Use default location'. Below it, a 'Location:' label followed by a text field containing '/home/stud2016/6gwizd/d/eclipse-workspace/Example2\_ServletsDBCo' and a 'Browse...' button.
- Target runtime:** A dropdown menu showing 'Apache Tomcat v9.0' and a 'New Runtime...' button.
- Dynamic web module version:** A dropdown menu showing '3.1'.
- Configuration:** A dropdown menu showing '<custom>' and a 'Modify...' button. Below it, a hint text: 'Hint: Get started quickly by selecting one of the pre-defined project configurations.'
- EAR membership:** A section with an unchecked checkbox 'Add project to an EAR'. Below it, an 'EAR project name:' label followed by a text field containing 'EAR' and a 'New Project...' button.
- Working sets:** A section with an unchecked checkbox 'Add project to working sets' and a 'New...' button. Below it, a 'Working sets:' label followed by a dropdown menu and a 'Select...' button.

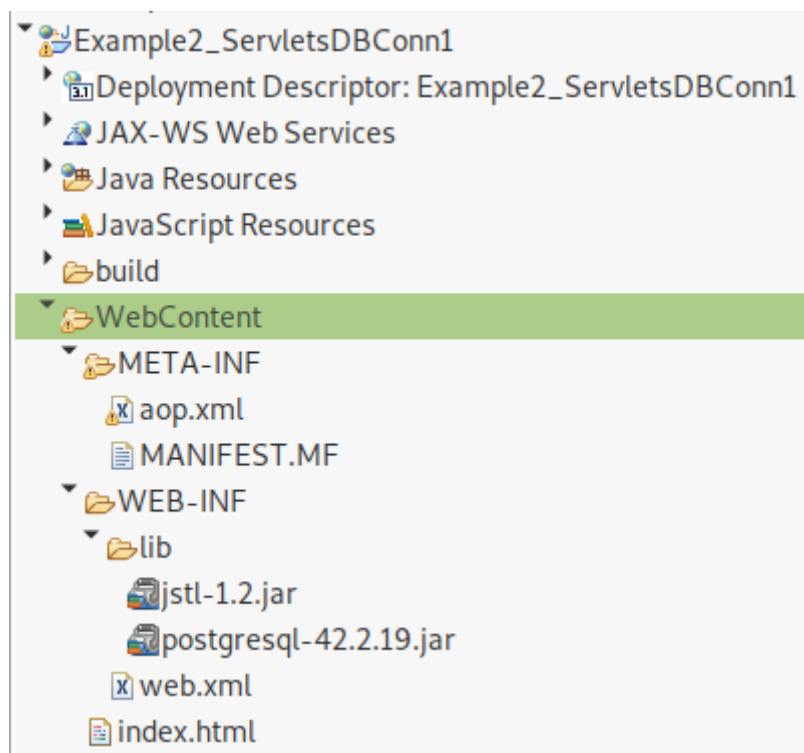
At the bottom of the dialog, there is a help icon (question mark in a circle) and four buttons: '< Back', 'Next >', 'Cancel', and a green 'Finish' button.



W katalogu **Example2\_ServletsDBConn/WebContent** znajdują się pliki, które należy skopiować do katalogu **WebContent** w projekcie.

Dodajemy przez to między innymi:

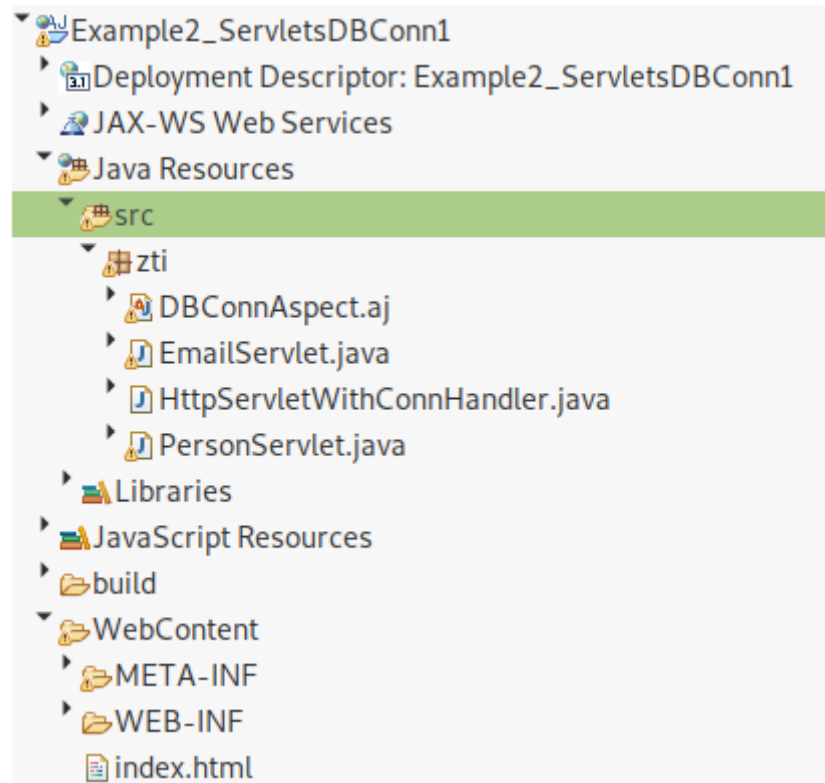
- Biblioteki **postgresql**, **jstl** oraz dotyczące aspektów: **aspectj** i **aspectjweaver**.  
Jeżeli nie masz sklonowanego repozytorium na maszynie wirtualnej to biblioteki postgresql i jstl można skopiować z poprzednich projektów robionych na laboratoriach, a biblioteki do aspektów pobieramy stąd:  
<https://mvnrepository.com/artifact/org.aspectj/aspectjrt/1.9.6>  
<https://mvnrepository.com/artifact/org.aspectj/aspectjweaver/1.9.6>
- Plik **aop.xml** zawierający informacje o aspekcie, który zaraz dodamy.
- Plik **index.html**, w którym są linki do dwóch servletów, które też zaraz dodamy.



Następnie dodajemy w projekcie do **src** paczkę **zti** i kopiujemy pliki:

- **HttpServletWithConnHandler.java**
- **PersonServlet.java**
- **EmailServlet.java**
- **DBConnAspect.aj**

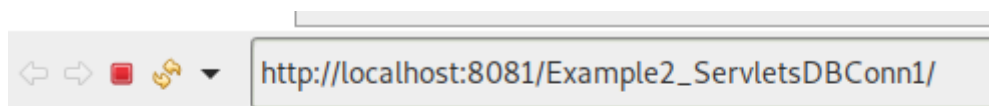
Pliki źródłowe znajdują się w repozytorium w katalogu **Example2\_ServletsDBConn1/src**.



W pliku **DBConnAspect.aj** należy wypełnić swoje dane do połączenia z bazą danych:

```
1 package zti;
2
3 import java.sql.DriverManager;
4
5
6 public aspect DBConnAspect {
7
8     pointcut doGetPersonServletPointcut(HttpServletWithConnHandler servlet) : execution
9
10    before(HttpServletWithConnHandler servlet) : doGetPersonServletPointcut(servlet) {
11
12        System.out.println("Advice BEFORE");
13
14        try{
15            Class.forName("org.postgresql.Driver");
16            String url = "jdbc:postgresql://[redacted].db.elephantsql.com:5432/[redacted]";
17            String username = "[redacted]";
18            String password = "[redacted]";
19            System.out.println();
20            servlet.conn = DriverManager.getConnection(url, username, password);
21        } catch (Exception e){}
22    }
23
24    after(HttpServletWithConnHandler servlet) : doGetPersonServletPointcut(servlet) {
25        System.out.println("Advice AFTER");
26
27        try{
28            servlet.conn.close();
29        } catch (Exception e){}
30    }
31 }
32
```

Aplikację można już uruchomić na serwerze Tomcat.



1. [PersonServlet](#)
2. [EmailServlet](#)

## The data from the PostgreSQL database

ID	Firstname	Lastname	City
7	Robur	Lewandowski	Monachium
9	Lukasz	Fabianski	Londyn
10	Wojciech	Szczesny	Turyn
12	Jakub	Moder	Brighton
11	Jan	Bednarek	Southampton
15	Jakas	Baba	
14	Adam	Babacki	Warszawa
16	Dawid	Gxxxxx	Krakow
8	Mateusz	Klich	Leeds

## The data from the PostgreSQL database

ID	Email
7	null
9	null
10	null
12	null
11	jb@mail.com.pl
15	a@b.c
14	ab@mail.com
16	xxx@xxx.xx
8	klich@klich.com

Tomcat v9.0 Server at localhost [Apache Tomcat] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (

Advice BEFORE

Advice AFTER

Advice BEFORE

Advice AFTER

Advice BEFORE

Advice AFTER

## Przykład 3.

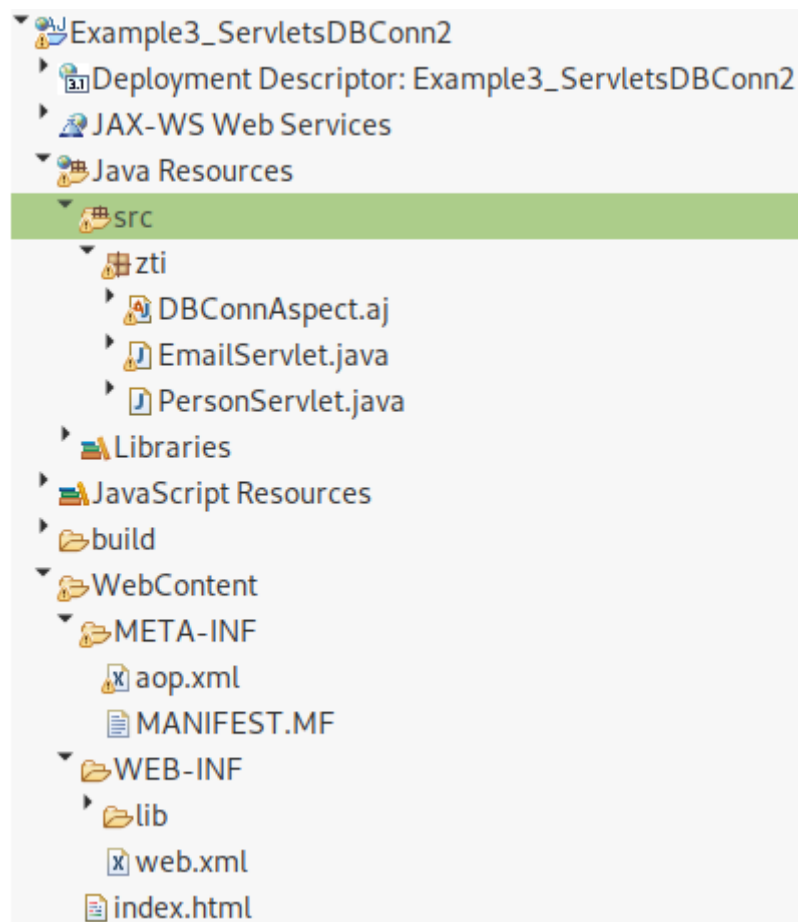
Przykład trzeci jest podobny do przykładu drugiego, dlatego najlepiej skopiować projekt przedstawiony w drugim przykładzie zmieniając nazwę.

Następnie usuwamy wszystkie pliki z paczki **zti** i dodajemy pliki albo zastępujemy te już istniejące:

- **PersonServlet.java**
- **EmailServlet.java**
- **DBConnAspect.aj**

Tym razem korzystając z plików źródłowych z katalogu **Example3\_ServletsDBConn2/src**.

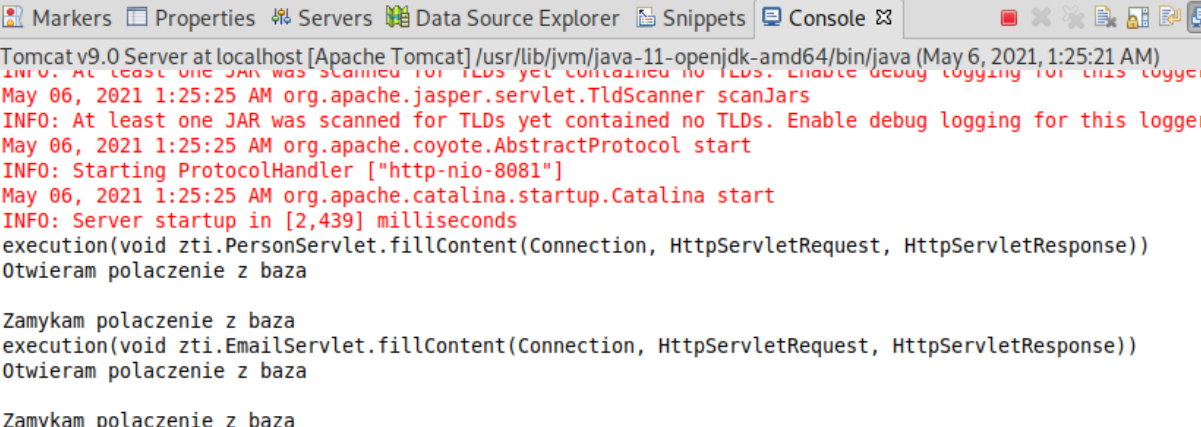
Zawartość **WebContent** jest taka sama.



Proszę nie zapomnieć o dodaniu danych do połączenia z bazą danych.

```
1 package zti;
2
3 import java.sql.DriverManager;
4
5
6
7
8
9 public aspect DBConnAspect {
10
11     pointcut doGetPersonServletPointcut(Connection conn, HttpServletRequest request, HttpSe
12         execution(* fillContent(Connection, HttpServletRequest, HttpServletResponse)) && args(c
13
14
15     void around(Connection conn, HttpServletRequest request, HttpServletResponse response) :
16
17         System.out.println(thisJoinPoint);
18
19         System.out.println("Otwieram polaczenie z baza");
20
21         try{
22             Class.forName("org.postgresql.Driver");
23             String url = "DATABASE_URL";
24             String username = "DATABASE_USERNAME";
25             String password = "DATABASE_PASSWORD";
26             System.out.println();
27             conn = DriverManager.getConnection(url, username, password);
28         } catch (Exception e){}
29
30         proceed(conn, request, response);
31
32         System.out.println("Zamykam polaczenie z baza");
33
34         try{
35             conn.close();
36         } catch (Exception e){}
37     }
38 }
39
```

Jeżeli skopiowano cały projekt z przykładu drugiego to ścieżka w URLu zostanie taka sama, aplikacja działa identycznie, ale generuje przy połączeniu trochę inne komunikaty na konsoli. Z tego powodu proszę upewnić się, że uruchomiono przykład trzeci, a nie drugi.



```
Tomcat v9.0 Server at localhost [Apache Tomcat] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (May 6, 2021, 1:25:21 AM)
INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger
May 06, 2021 1:25:25 AM org.apache.jasper.servlet.TldScanner scanJars
INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger
May 06, 2021 1:25:25 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-nio-8081"]
May 06, 2021 1:25:25 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in [2,439] milliseconds
execution(void zti.PersonServlet.fillContent(Connection, HttpServletRequest, HttpServletResponse))
Otwieram polaczenie z baza

Zamykam polaczenie z baza
execution(void zti.EmailServlet.fillContent(Connection, HttpServletRequest, HttpServletResponse))
Otwieram polaczenie z baza

Zamykam polaczenie z baza
```

# Posłowie

Przy przygotowywaniu seminarium i przykładów korzystałem z kilku źródeł, jakby ktoś chciał zgłębić bardziej temat to podaję linki do kilku stron.

O programowaniu w AspectJ z przykładami:

<https://o7planning.org/10257/java-aspect-oriented-programming-with-aspectj>

<https://www.baeldung.com/tag/aspectj/>

- Wdrożenie aplikacji aspektowej (z użyciem Mavena):

<https://www.baeldung.com/aspectj>

Porównanie Spring AOP i AspectJ:

- <https://www.baeldung.com/spring-aop-vs-aspectj>