

AspectJ - Ćwiczenia

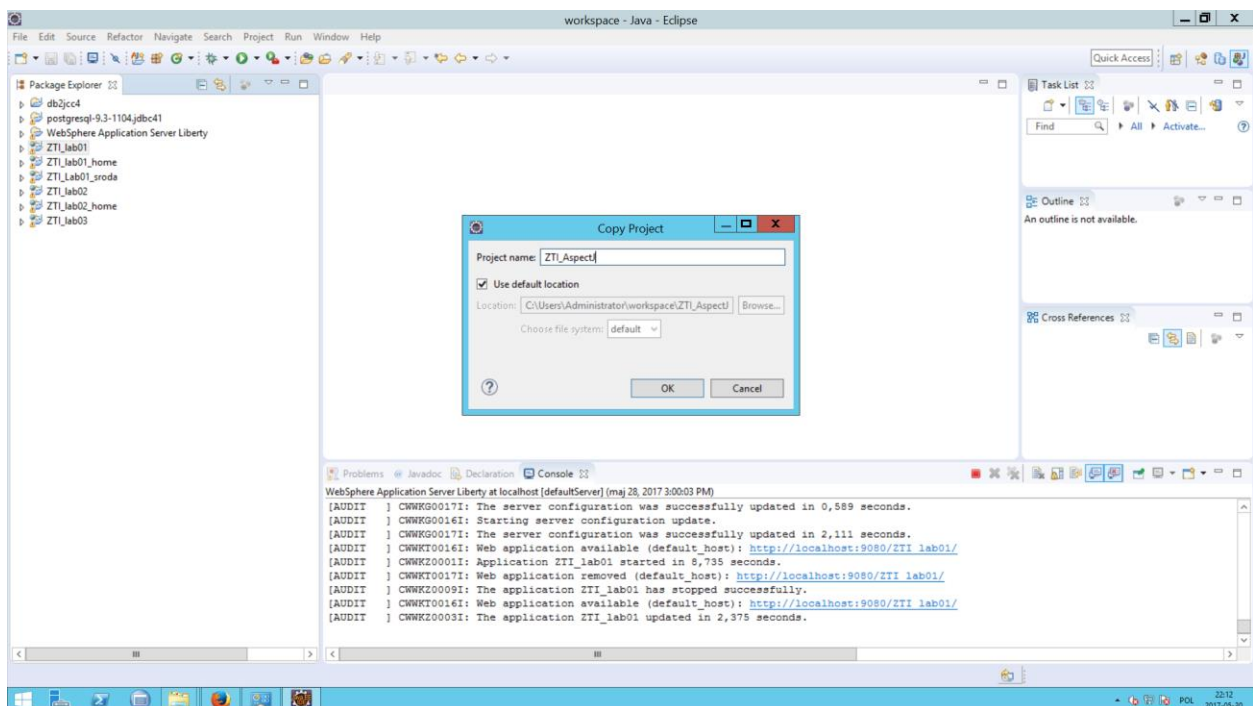
Zaawansowane Technologie Internetowe

Agata Wójcik, Mateusz Winiarski

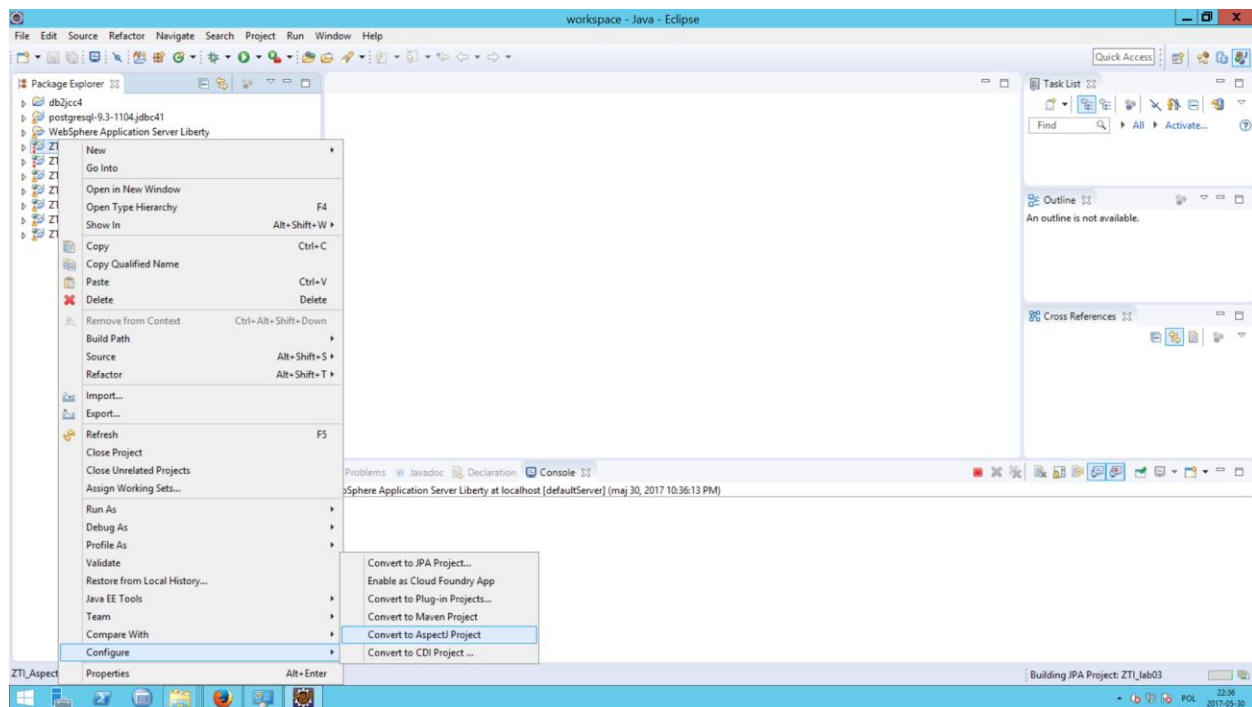
1. Przygotowanie środowiska

2. Zadanie 1. – Auto Logger

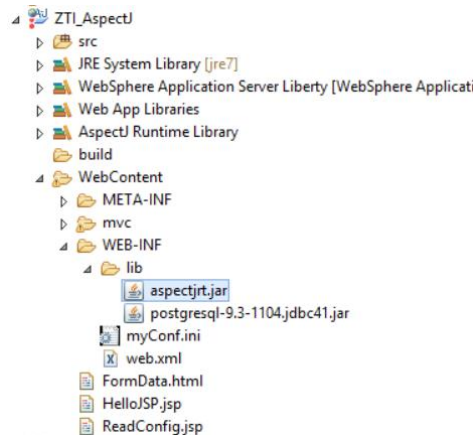
Najpierw skopiuj jedno ze swoich działających poprzednich zadań (w moim wypadku jest to kod z laboratorium 1.). Nadaj mu nową nazwę.



Po tym musisz przekształcić projekt w projekt AspectJ. Żeby to zrobić kliknij prawym przyciskiem myszy na projekcie, rozwiń menu **Configure** i wybierz opcję **Convert to AspectJ Project**.



Teraz, żeby móc używać języka AspectJ musisz jedynie dołączyć do projektu plik aspectjrt.jar. Przeciągnij go do katalogu [nazwa projektu]/WebContent/WEB-INF/lib/. Po tym Twój projekt powinien wyglądać mniej więcej jak na zrzucie ekranu poniżej.



W tym momencie masz gotowy projekt, żeby użyć w nim AspectJ.

Teraz stworzymy jedynie dwa pliki, żeby zaimplementować automatyczne logowanie wykonywanych metod w naszym projekcie.

Najpierw stworzymy nową klasę o nazwie **Logger** w moim wypadku pakiecie **ZTI_Lab01**. Jej kod jest niezmiernie prosty i zamieszczony poniżej.

```
package ZTI_Lab01;

public class Logger {
    public static void entry(String message){
```

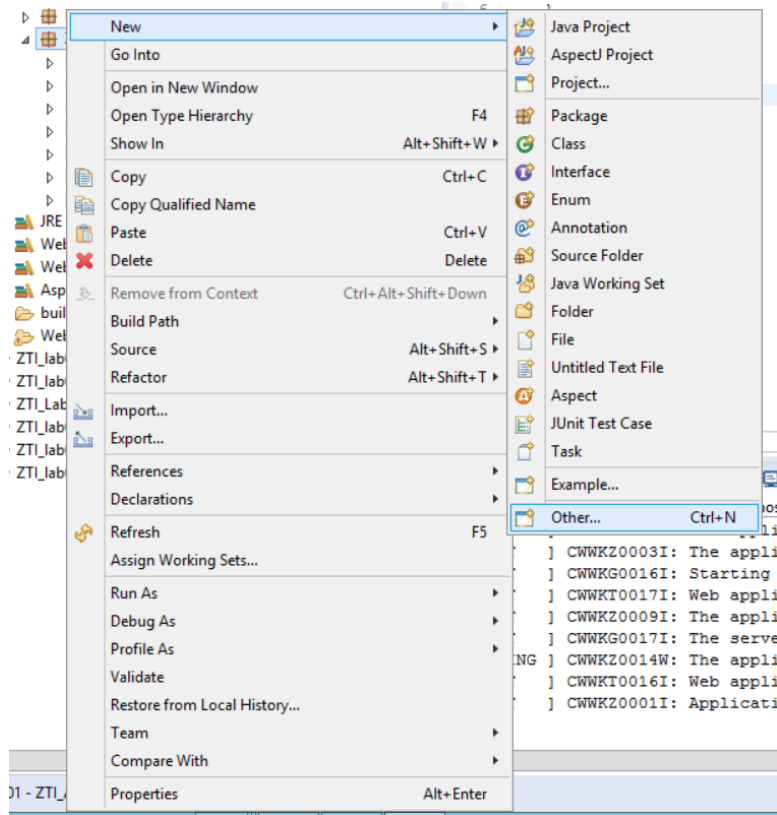
```

        System.out.println "[" + i + "] Entering method " + message);
    }

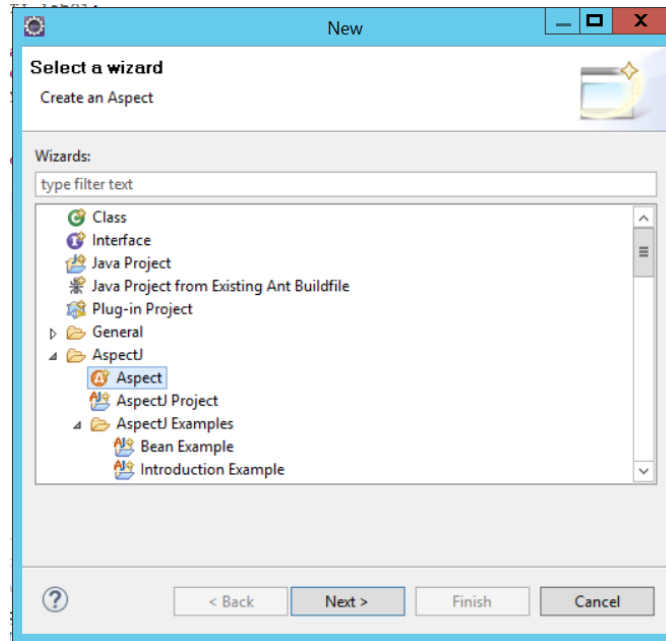
    static int i = 0;
}

```

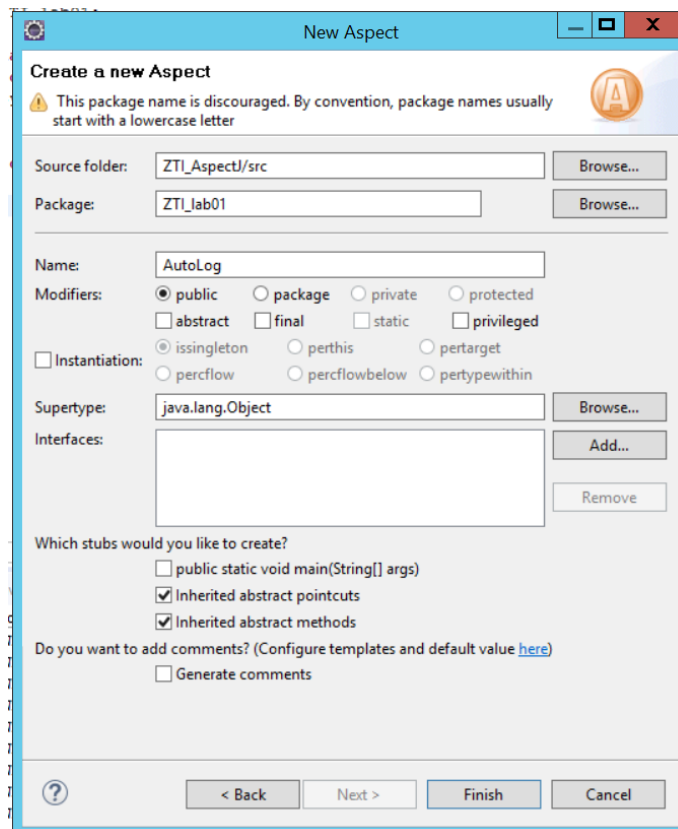
Następnie w tym samym pakiecie dodamy aspekt AutoLog. Kliknij na pakiet prawym przyciskiem myszy, wybierz New, a następnie Other.



Następnie w rozwin AspectJ, wybierz Aspect i kliknij Next.



Wpisz wspomnianą wyżej nazwę, po czym kliknij Finish.



Kod aspektu zamieszczony jest poniżej.

```
package ZTI_Lab01;

public aspect AutoLog {
```

```

public static void Logger.exit(String message){
    System.out.println "[" + i + "] Exiting method " + message);
    ++i;
}

pointcut publicMethods() : execution(protected * ZTI_Lab01..*(..));

pointcut logObjectCalls() : execution(* Logger.*(..));

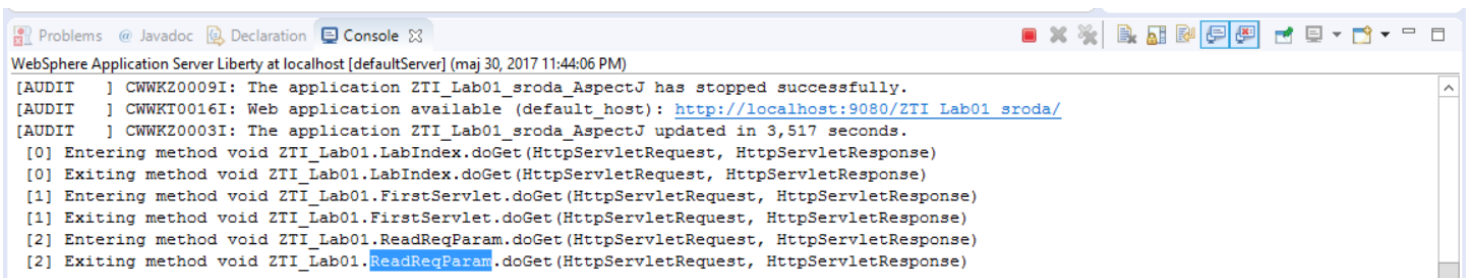
pointcut loggableCalls() : publicMethods() && ! logObjectCalls();

before() : loggableCalls(){
    Logger.entry(thisJoinPoint.getSignature().toString());
}

after() : loggableCalls(){
    Logger.exit(thisJoinPoint.getSignature().toString());
}
}

```

Następnie tak przygotowaną aplikację uruchamiamy poprzez Run As => Run On Server. Po jej odpaleniu w konsoli powinniśmy widzieć odpowiednie logi pojawiające się w momentach przechodzenia między różnymi stronami naszej aplikacji.



```

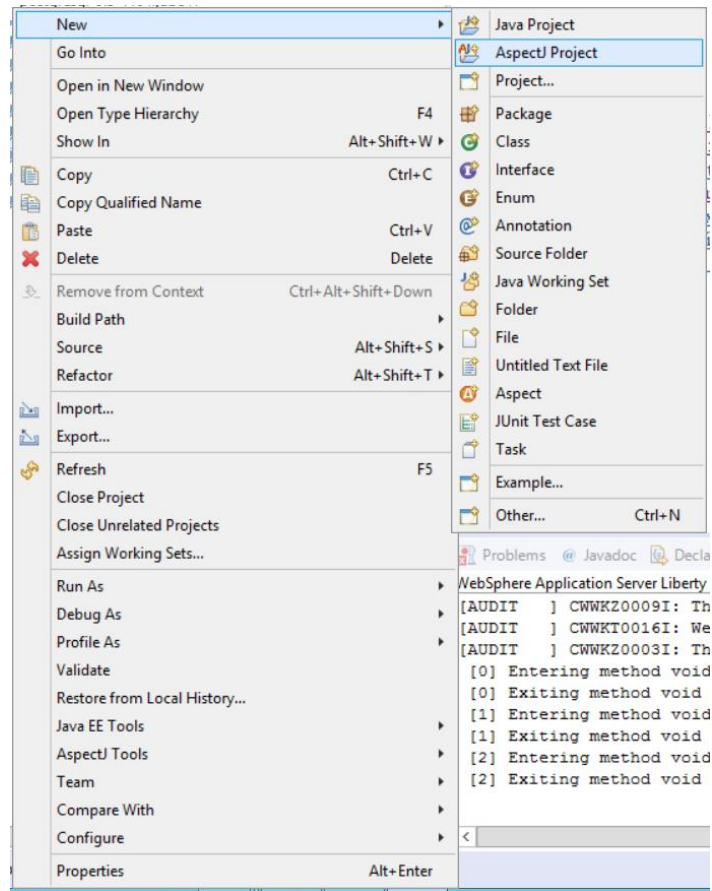
WebSphere Application Server Liberty at localhost [defaultServer] (maj 30, 2017 11:44:06 PM)
[AUDIT ] CWWKZ0009I: The application ZTI_Lab01_sroda_AspectJ has stopped successfully.
[AUDIT ] CWWKT0016I: Web application available (default_host): http://localhost:9080/ZTI_Lab01_sroda/
[AUDIT ] CWWKZ0003I: The application ZTI_Lab01_sroda_AspectJ updated in 3,517 seconds.
[0] Entering method void ZTI_Lab01.LabIndex.doGet(HttpServletRequest, HttpServletResponse)
[0] Exiting method void ZTI_Lab01.LabIndex.doGet(HttpServletRequest, HttpServletResponse)
[1] Entering method void ZTI_Lab01.FirstServlet.doGet(HttpServletRequest, HttpServletResponse)
[1] Exiting method void ZTI_Lab01.FirstServlet.doGet(HttpServletRequest, HttpServletResponse)
[2] Entering method void ZTI_Lab01.ReadReqParam.doGet(HttpServletRequest, HttpServletResponse)
[2] Exiting method void ZTI_Lab01.ReadReqParam.doGet(HttpServletRequest, HttpServletResponse)

```

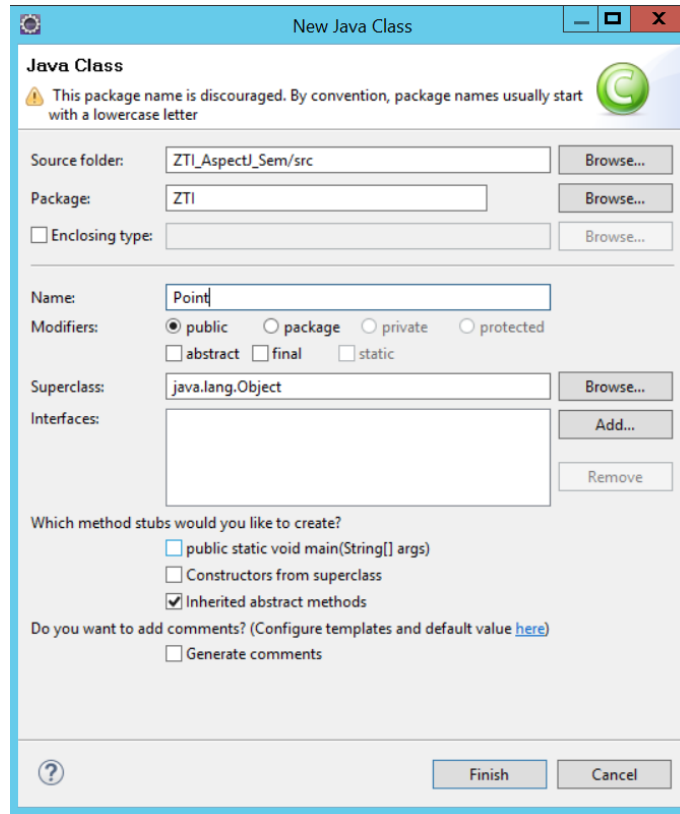
3. Zadanie 2.

Jest to dużo prostsze zadanie mające na celu pokazanie możliwości technologii już niekoniecznie w aplikacji webowej.

Tworzymy nowy projekt AspectJ.



A następnie w nim w pakiecie ZTI tworzymy klasę Point.



Której kod wygląda następująco.

```
package ZTI;

public class Point {
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public double x, y;
}
```

Następnie definiujemy w tym samym pakiecie nowy aspekt, którego kod jest następujący.

```
package ZTI;

import java.lang.Math;

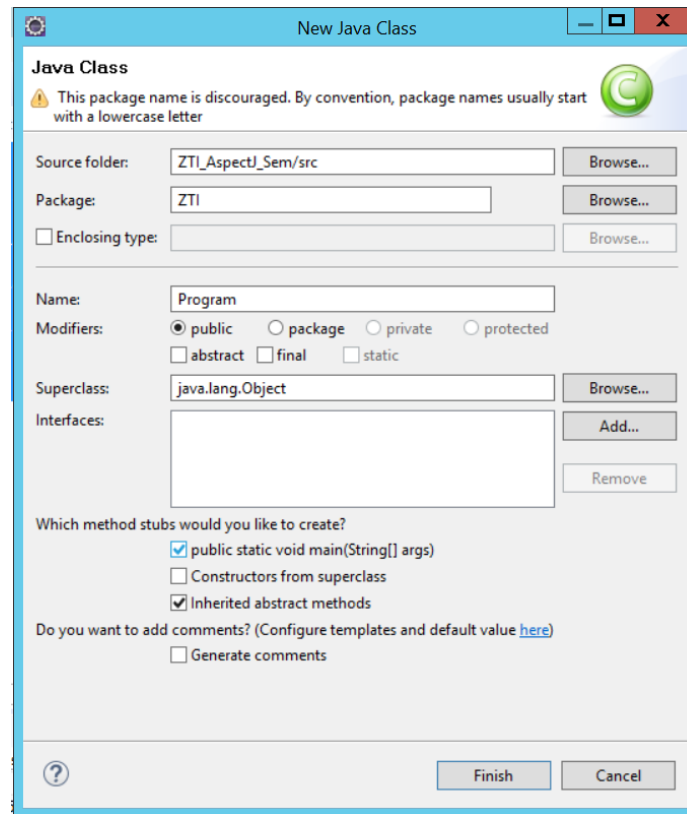
public aspect ComparablePoint {

    declare parents: Point implements Comparable;

    public int Point.compareTo(Object o) {
        return (int) (Math.sqrt(x * x + y * y) -
            Math.sqrt(((Point)o).x * ((Point)o).x + ((Point)o).y * ((Point)o).y));
    }
}
```

}

Na sam koniec tworzymy nową klasę, implementującą metodę main().



Kod tej klasy jest następujący.

```
package ZTI;

public class Program {

    public static void main(String[] args) {
        Point p1, p2;

        p1 = new Point(3,5);
        p2 = new Point(8,10);
        System.out.println("p1 == p2 : " + p1.compareTo(p2));

        p1 = new Point(3,5);
        p2 = new Point(3,5);
        System.out.println("p1 == p2 : " + p1.compareTo(p2));
    }
}
```

Aplikację uruchamiamy poprzez Run As => Java Application. Wynik w konsoli powinien być zgodny z poniższym zrzutem ekranu.

