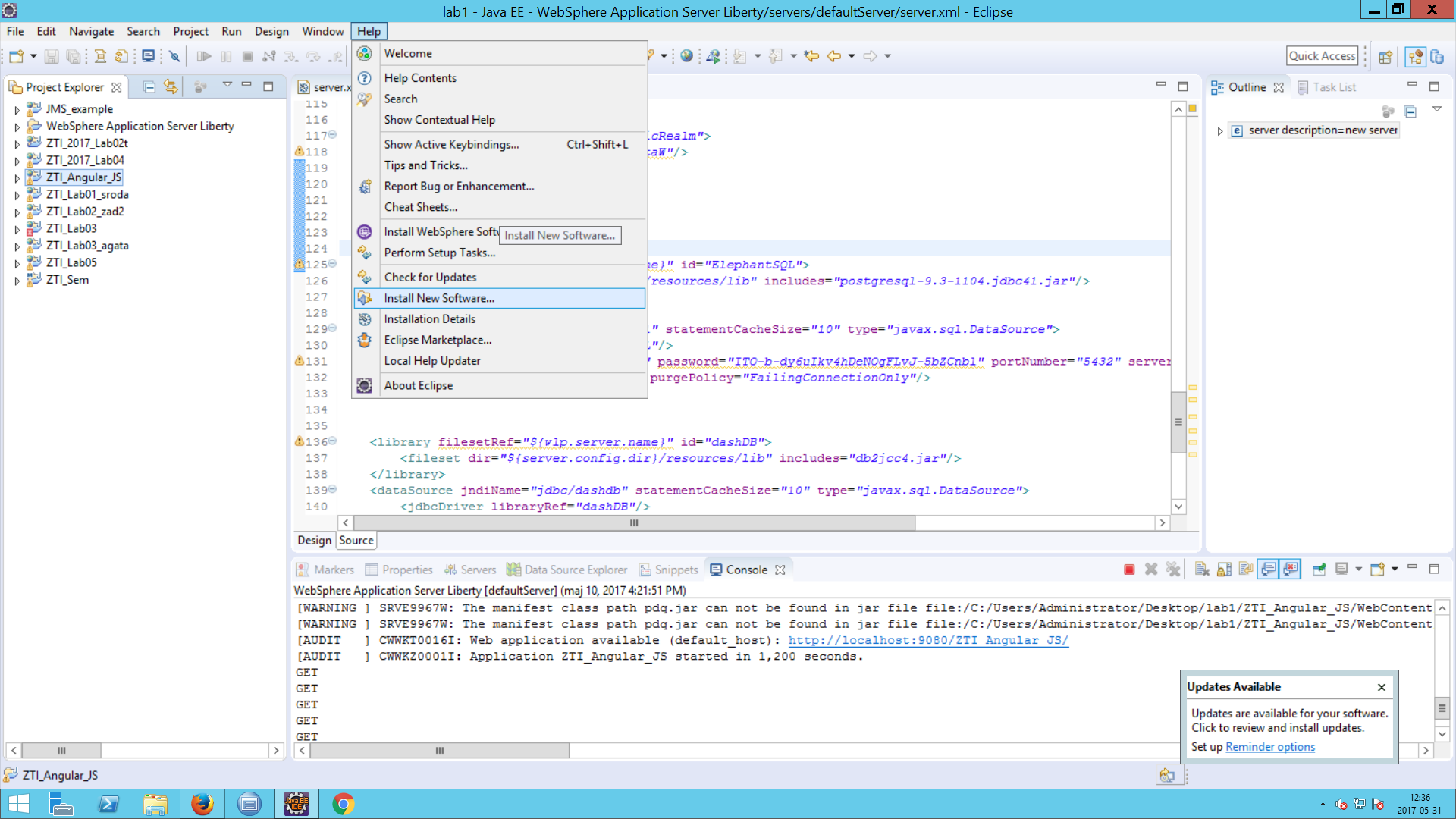
***AspectJ - Ćwiczenia***

Zaawansowane Technologie Internetowe

*Agata Wójcik, Mateusz Winiarski*

1. Przygotowanie środowiska

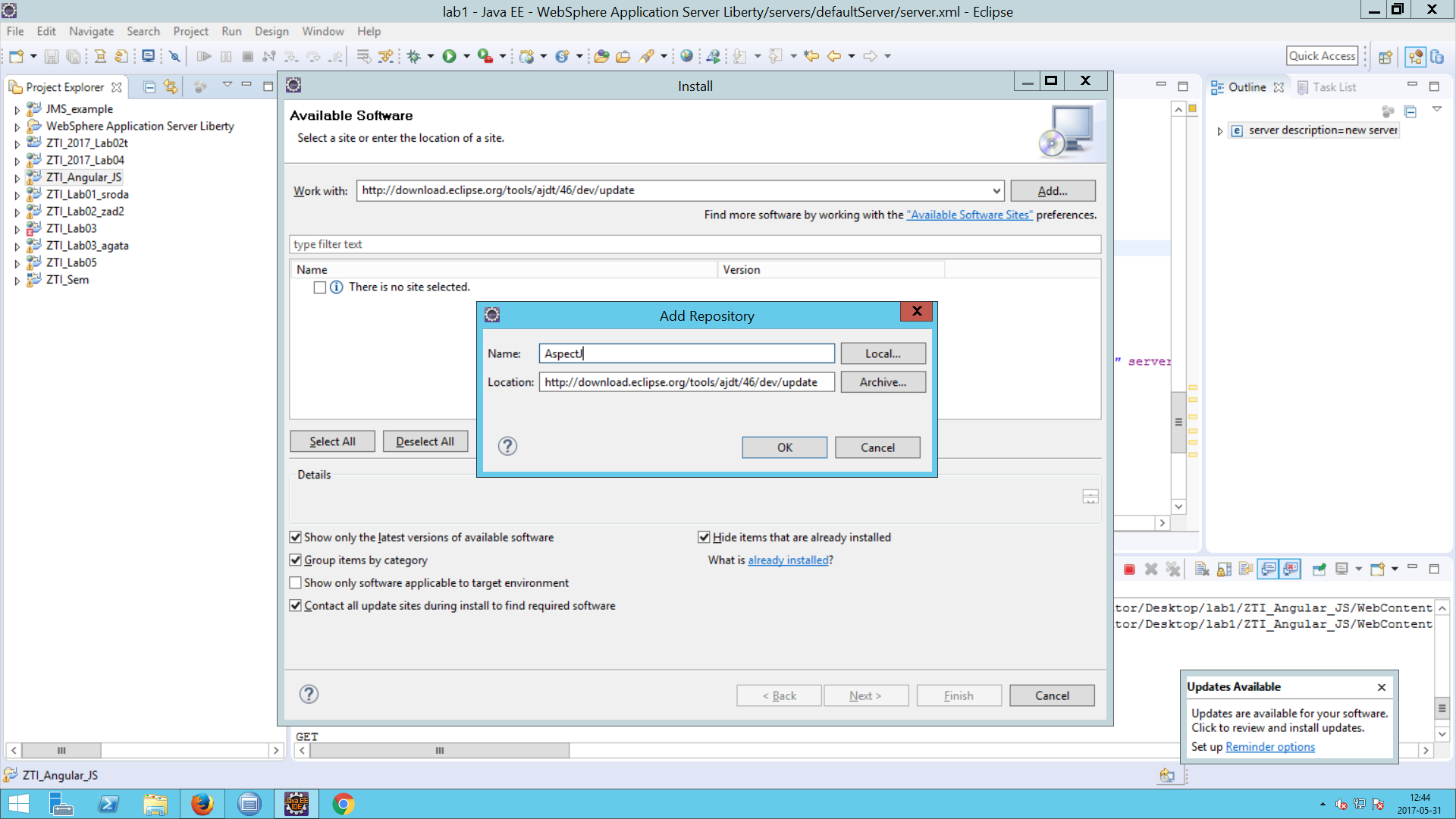
Żeby zainstalować dodatek do eclipse’a, który umożliwia tworzenie i kompilację projektów z użyciem języka AspectJ najpierw wybierz Help => Install New Software.



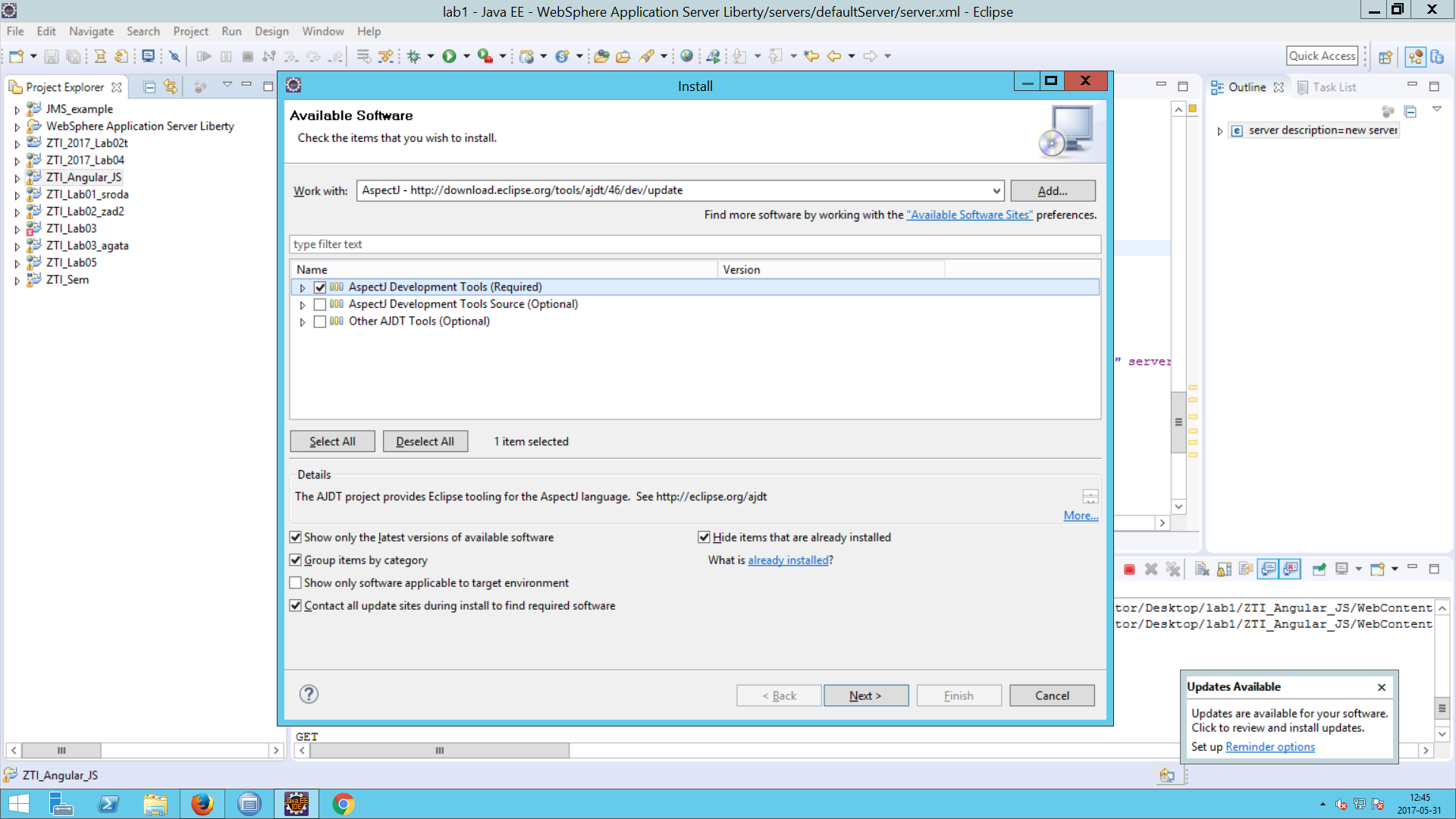
A następnie do pola Work With wklej poniższy adres a następnie kliknij przycisk Add. <http://download.eclipse.org/tools/ajdt/46/dev/update>



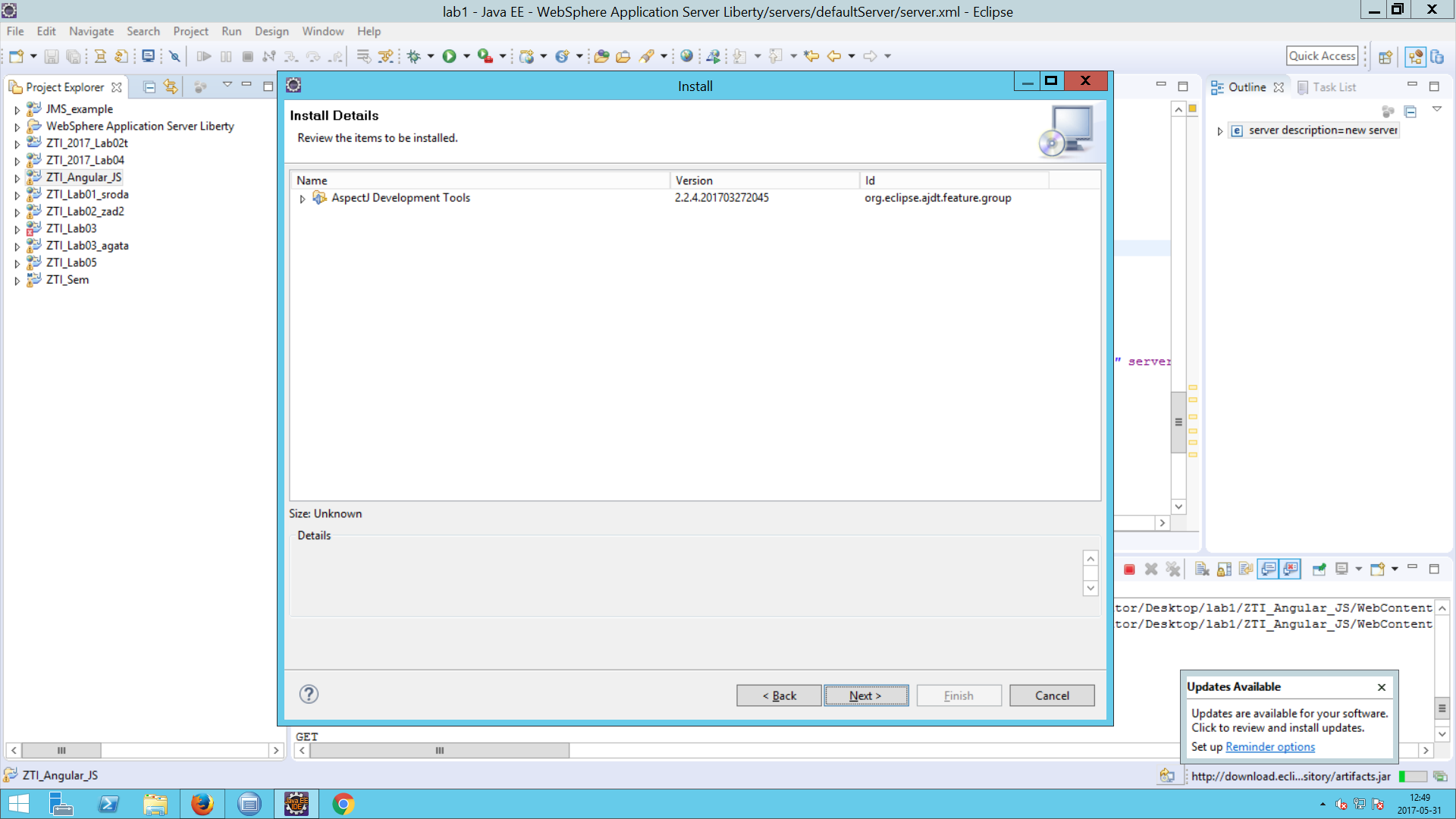
W oknie, które wyskoczy, w polu Name wpisz AspectJ.



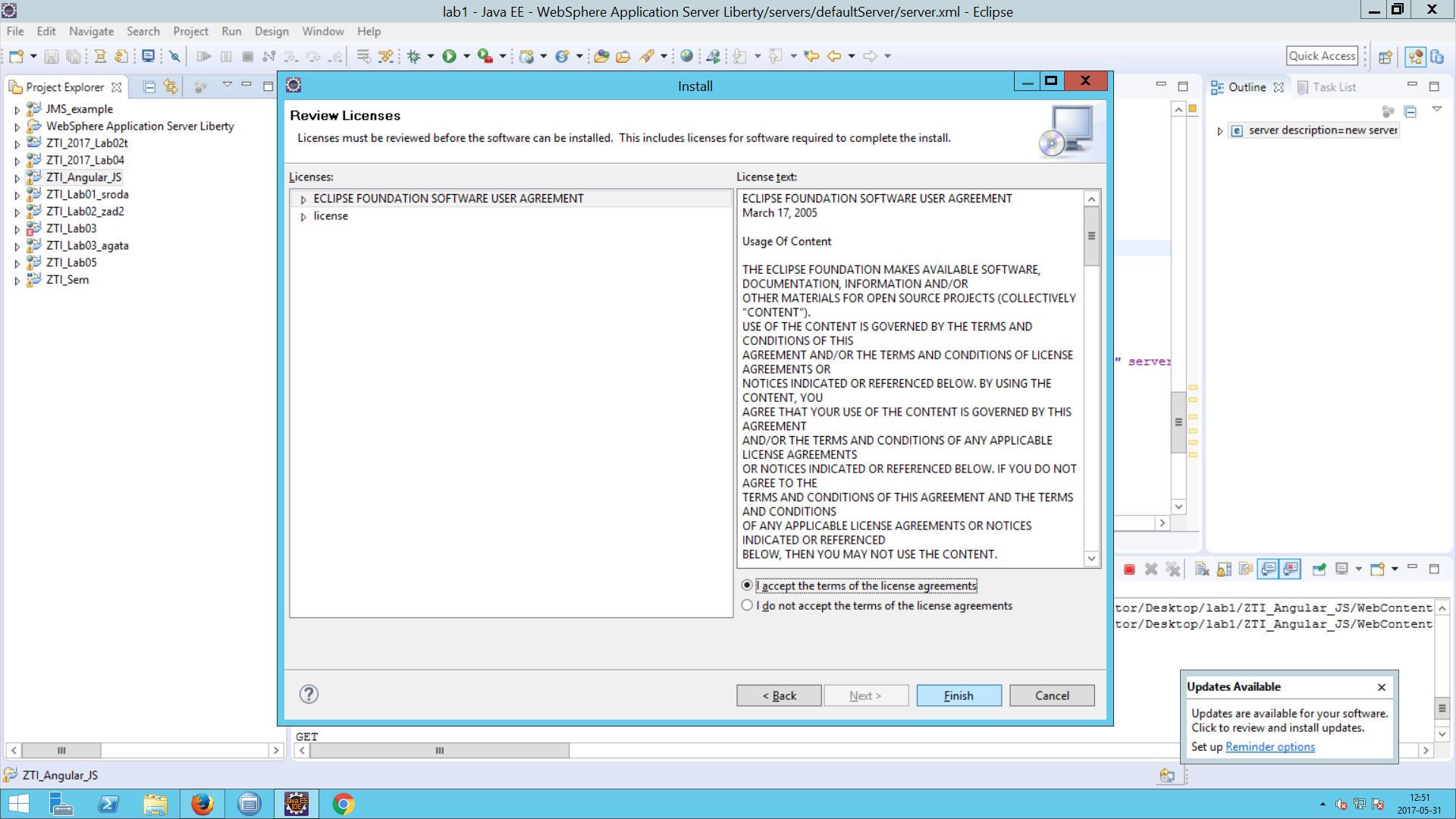
Po tym jak eclipse sprawdzi dostępne oprogramowanie spod tego adresu zaznacz AspectJ Development Tools (Required) i następnie kliknij Next.



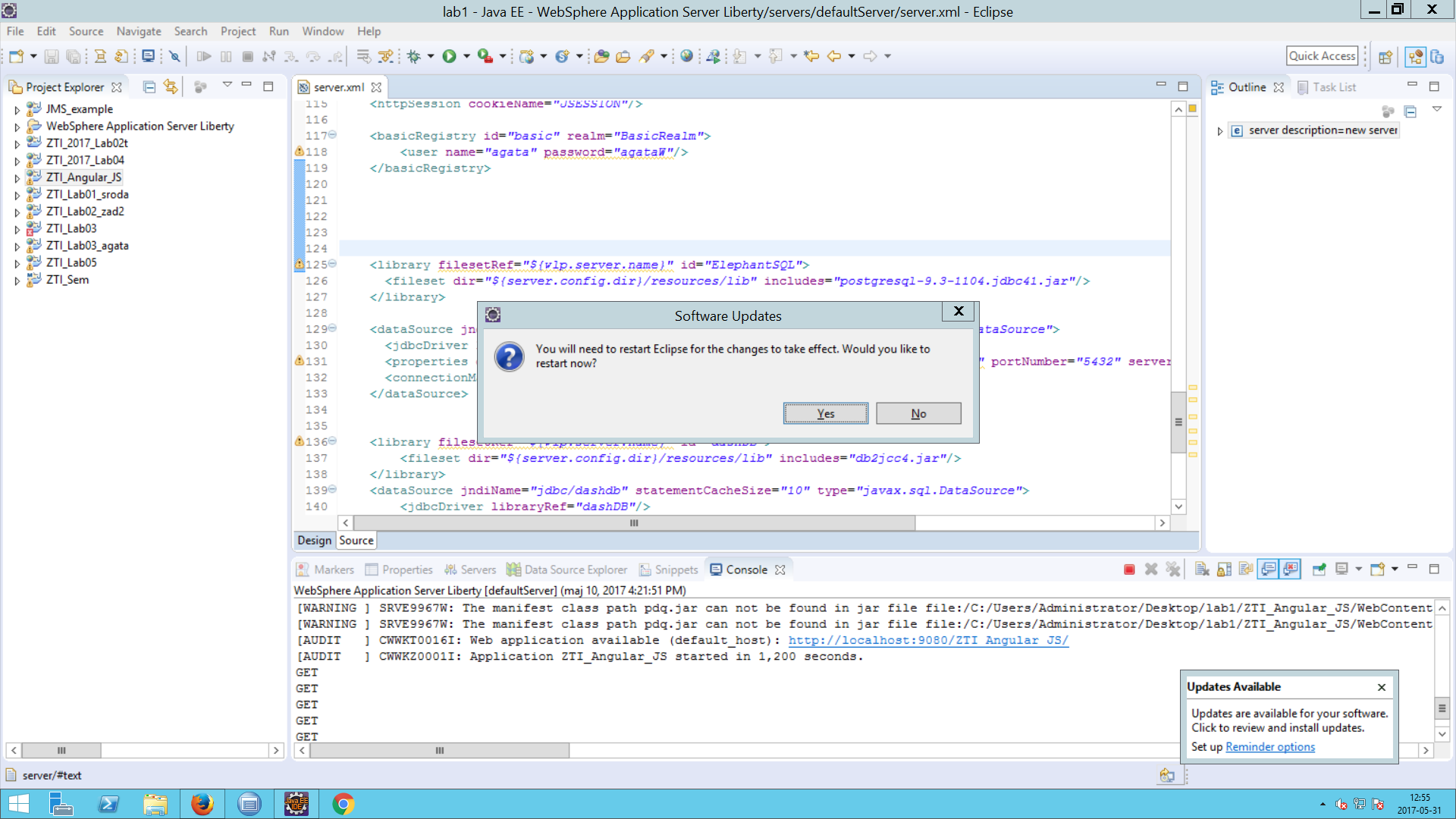
Eclipse poprosi Cię o sprawdzenie czy odpowiednie rzeczy będą zainstalowane. Jeśli okno wygląda jak to poniżej kliknij przycisk Next, w przeciwnym wypadku wróć do poprzedniego kroku klikają Back i zaznacz odpowiednie oprogramowanie.



W następnym kroku **musisz przeczytać licencję** tego dodatku. Po tym zaznacz, że zgadzasz się z nią i kliknij Finish.

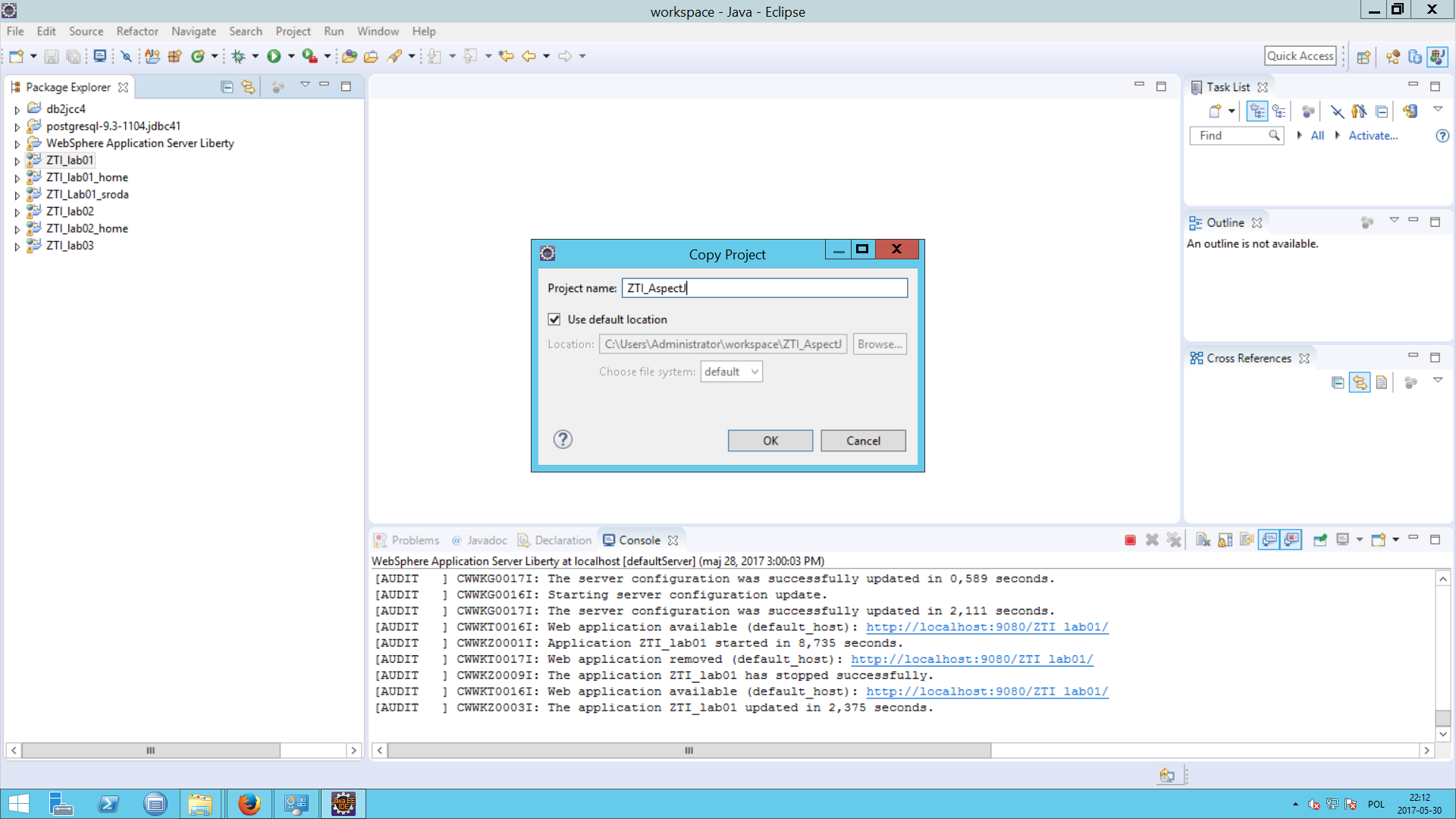


W tym momencie eclipse zainstaluje odpowiedni dodatek dzięki, któremu będziesz mógł korzystać ze wszystkich dobrodziejstw jakie dostarcza język AspectJ. Jednak, żeby móc w pełni cieszyć się z niego cieszyć eclipse musi zostać zrestartowany. Poprosi Cię o to sam, więc pozwól mu na to klikając Yes.

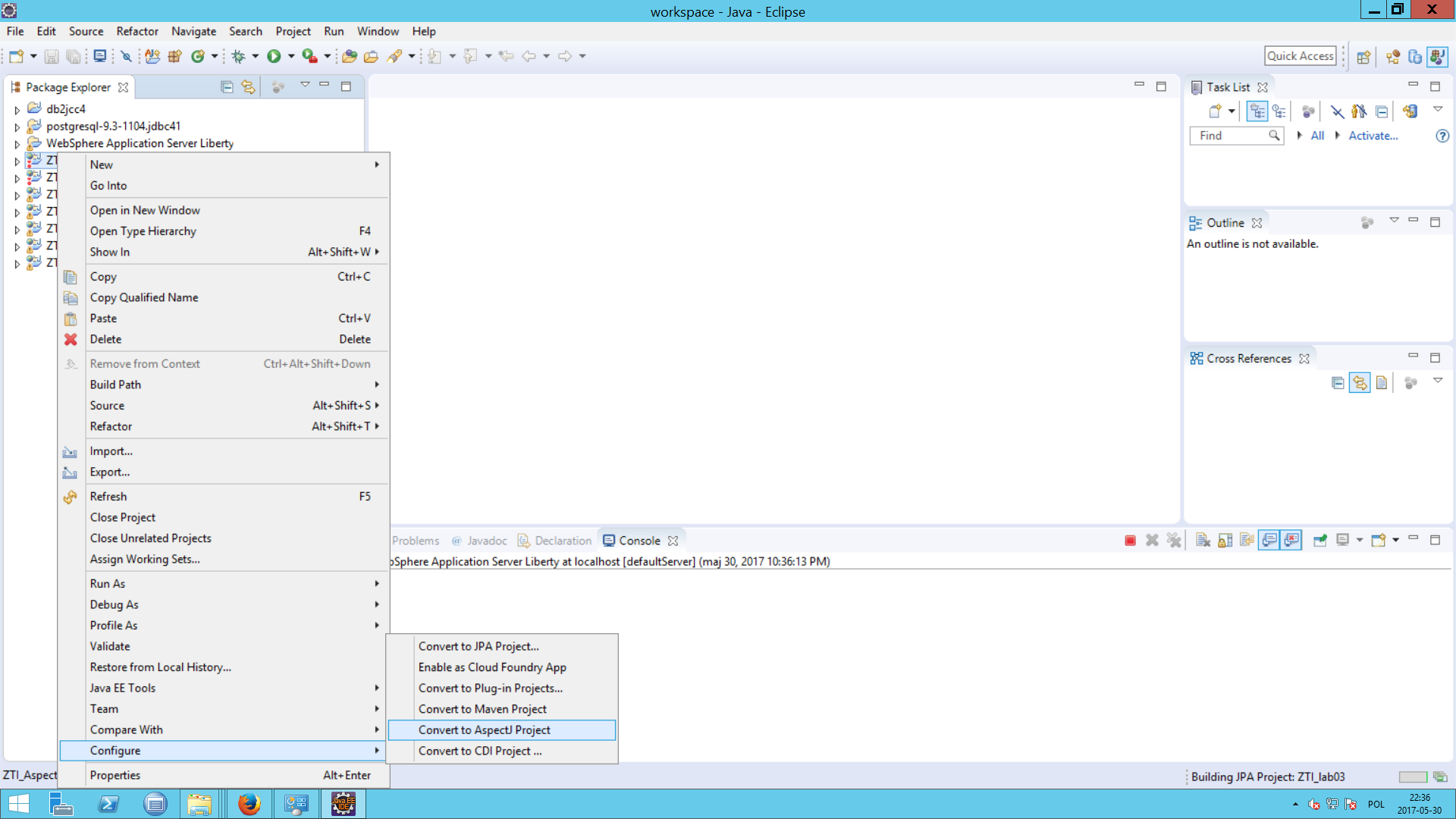


1. Zadanie 1. – Auto Logger

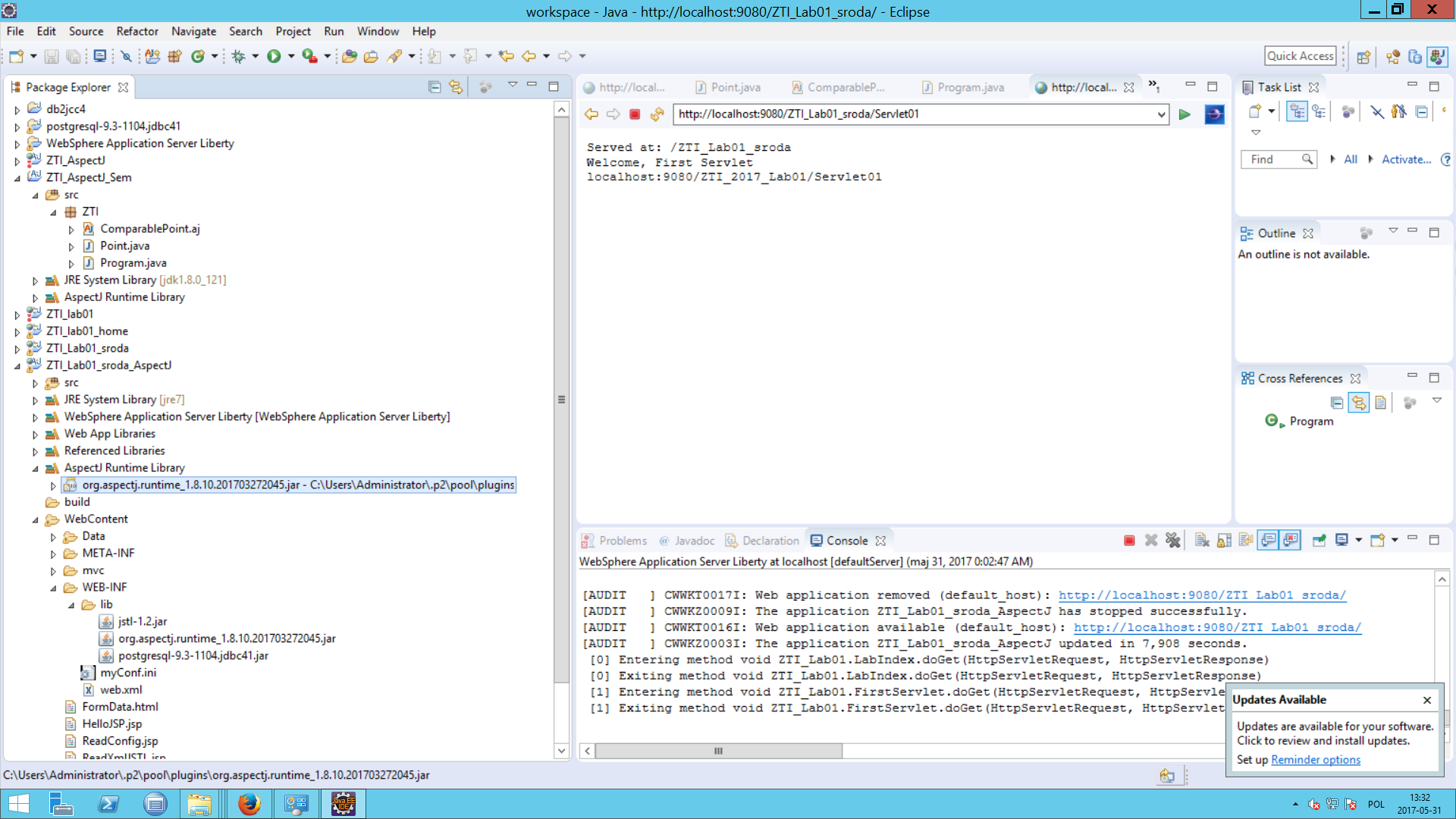
Najpierw skopiuj jedno ze swoich działających poprzednich zadań (w moim wypadku jest to kod z laboratorium 1.). Nadaj mu nową nazwę.



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



Teraz, żeby móc używać języka AspectJ musisz jedynie dołączyć do projektu plik org.aspectj.runtime\_1.8.10.201703272045.jar znajduje się on w folderze wskazywanym w AspectJ Runtime Library, w moim wypadku jest to: C:\Users\Administrator\.p2\pool\plugins\. 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){

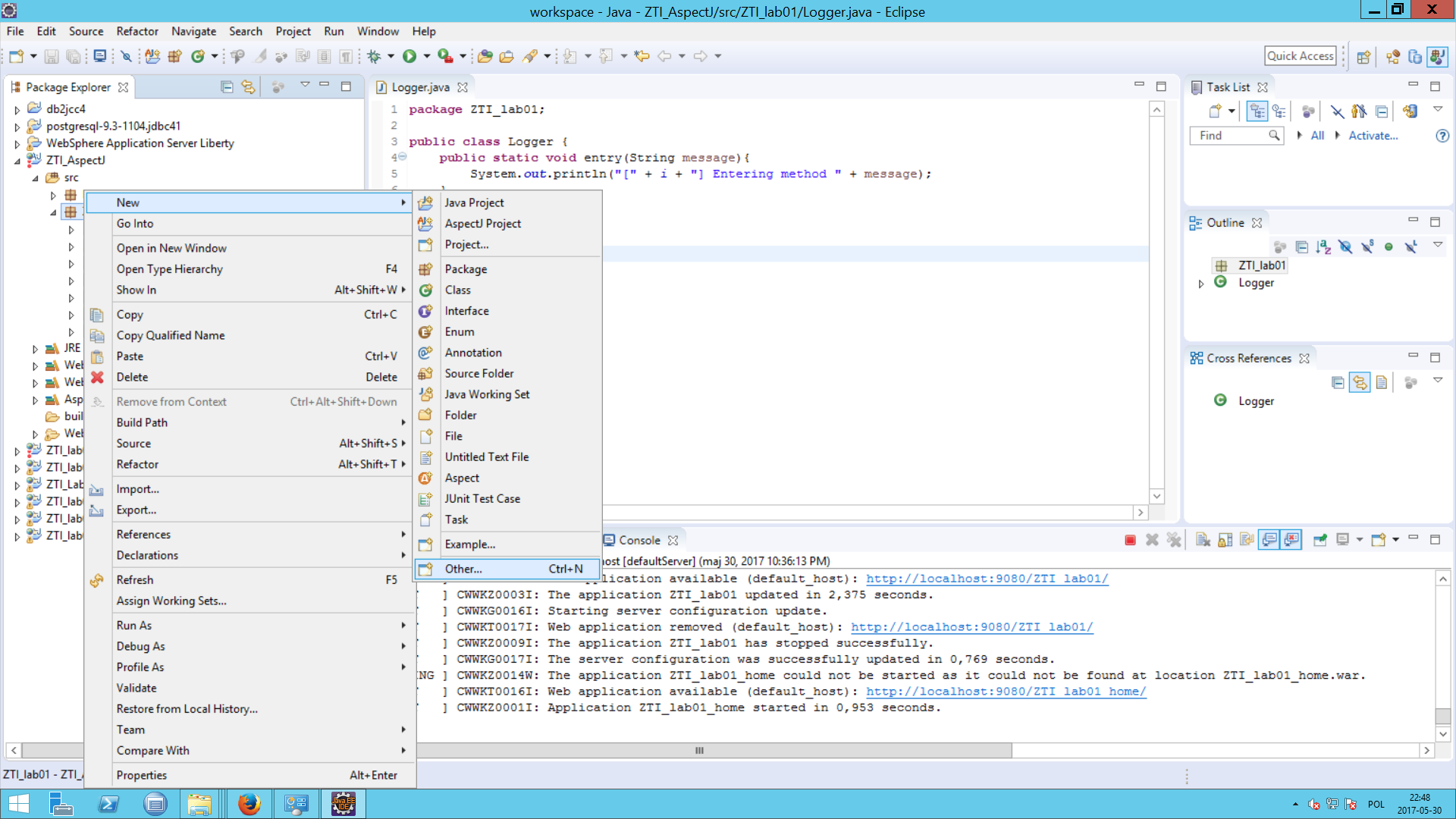
System.***out***.println("[" + *i* + "] Entering method " + message);

}

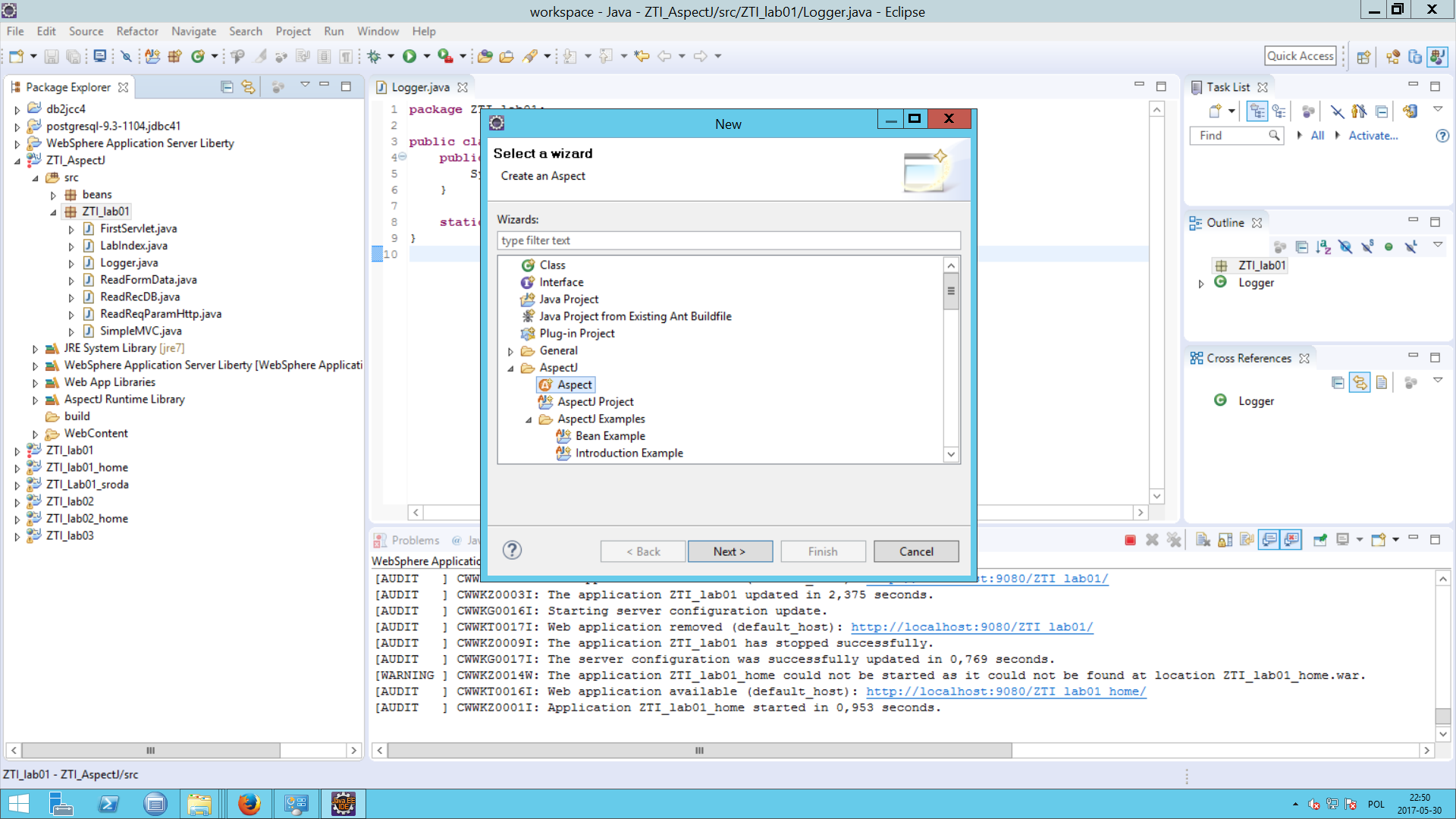
**static** **int** *i* = 0;

}

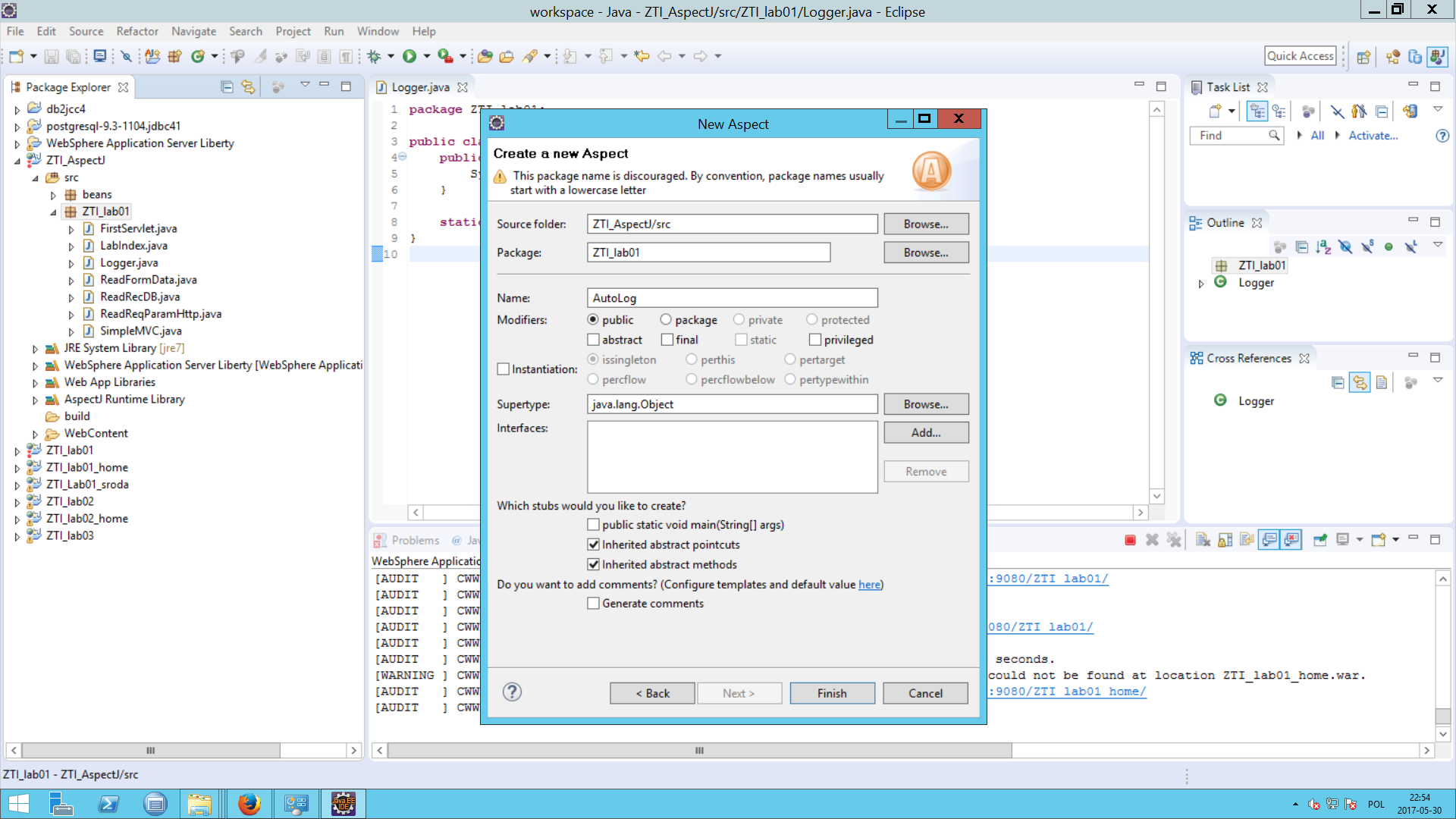
Następnie w tym samym pakiecie dodamy aspekt AutoLog. Kliknij na pakiet prawym przyciskiem myszy, wybierz New, a następnie Other.



Następnie w rozwiń 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());

}

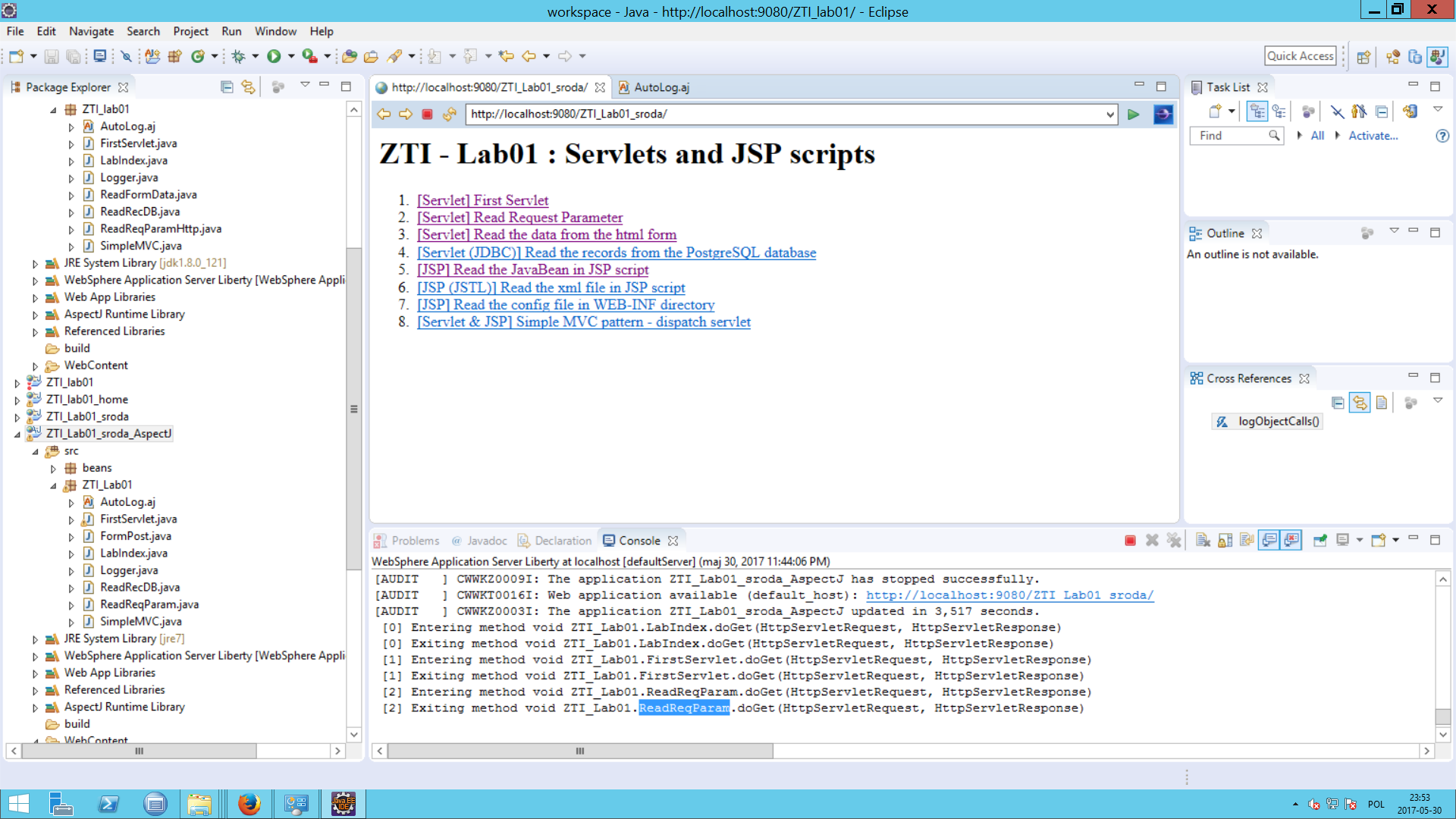
**after**() : loggableCalls(){

Logger.exit(**thisJoinPoint**.getSignature().toString());

}

}

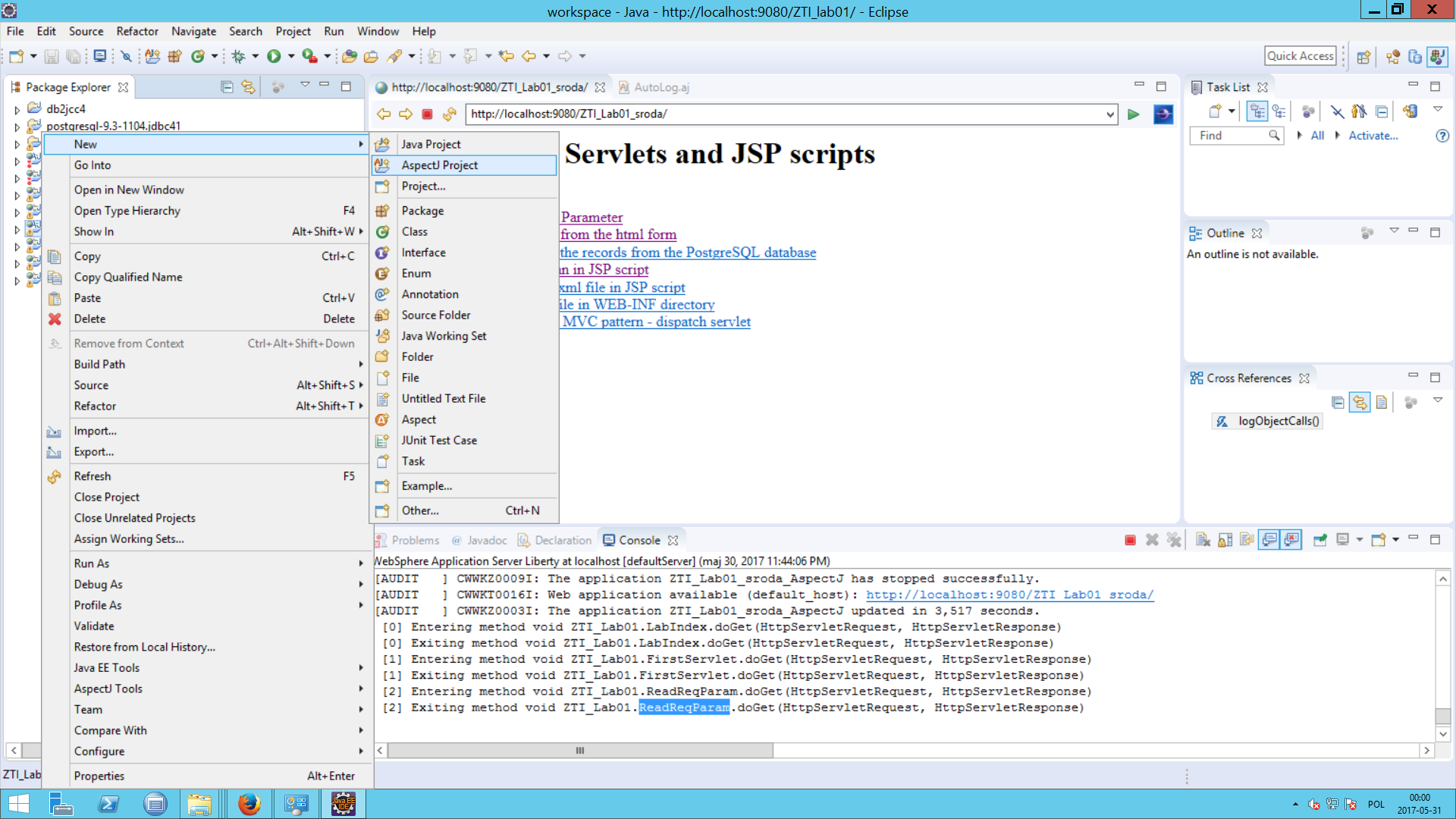
Następnie tak przygotowaną aplikację uruchamiany poprzez Run As => Run On Server. Po jej odpaleniu w konsoli powinniśmy widzieć odpowiednie logi pojawiające się w momentach przechodzenia miedzy różnymi stronami naszej aplikacji.



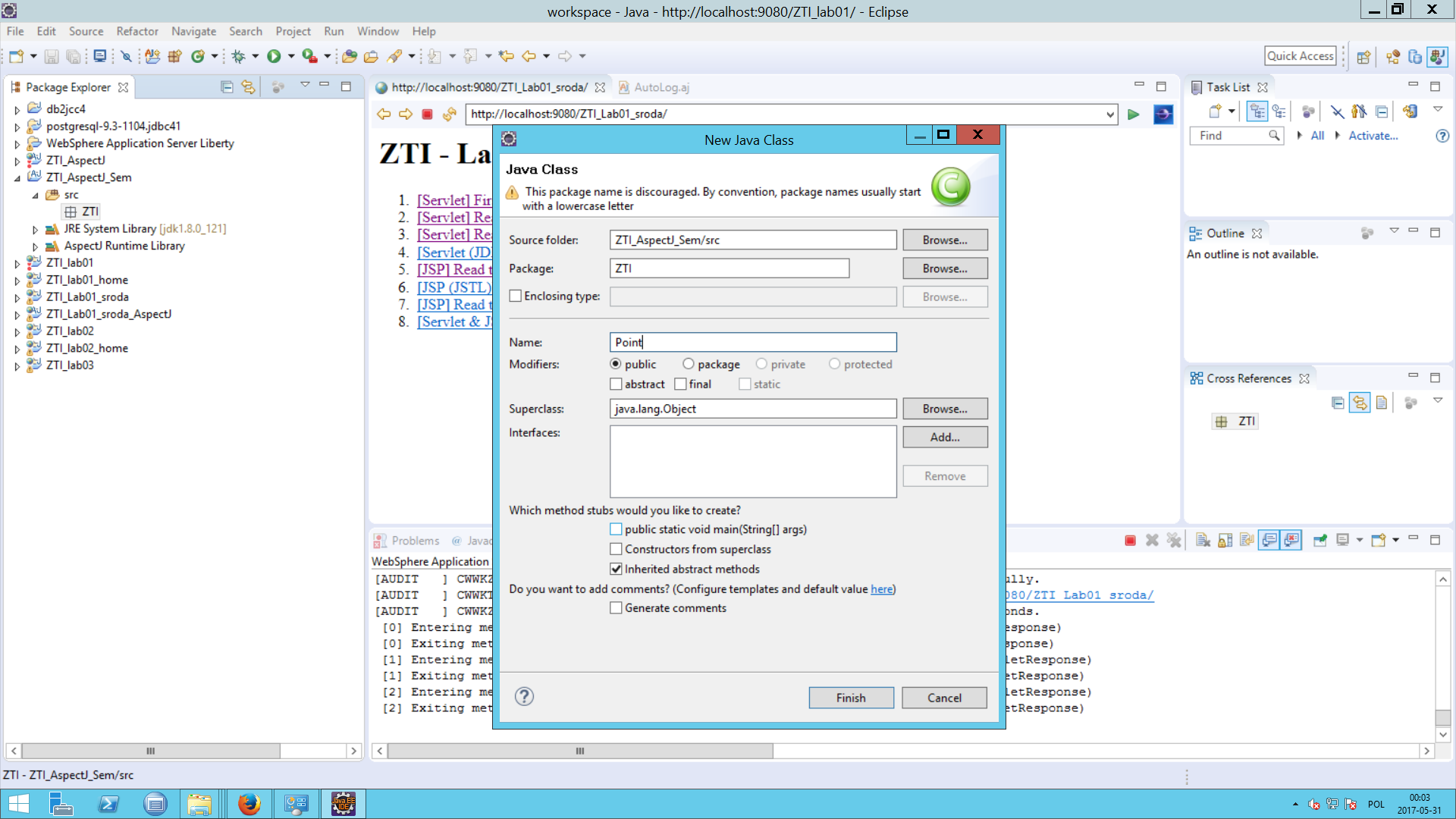
1. Zadanie 2.

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) {

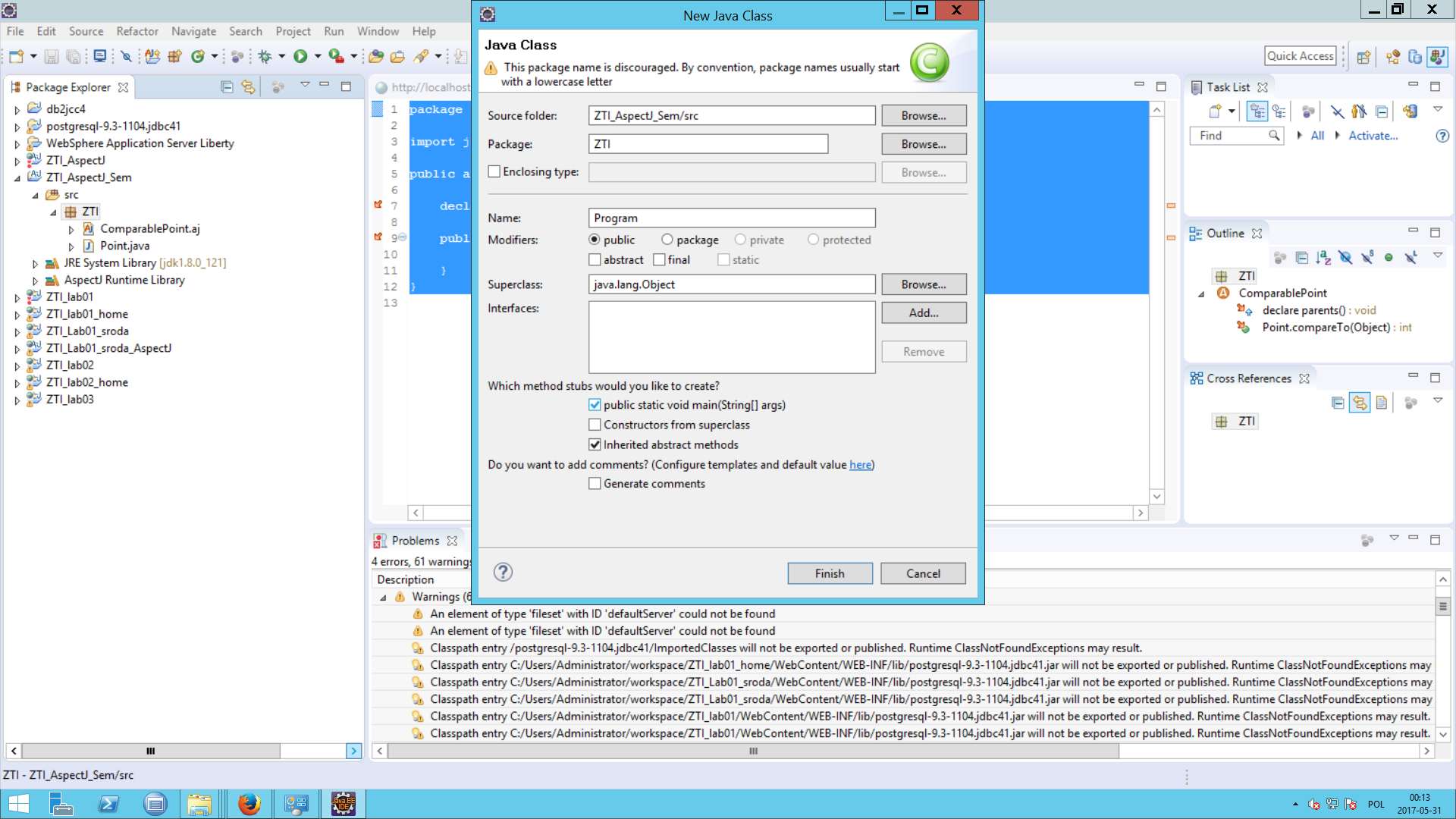
**return** (**int**) (Math.*sqrt*(x \* x + y \* y) -

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 Aplication. Wynik w konsoli powinien być zgodny z poniższym zrzutem ekranu.

