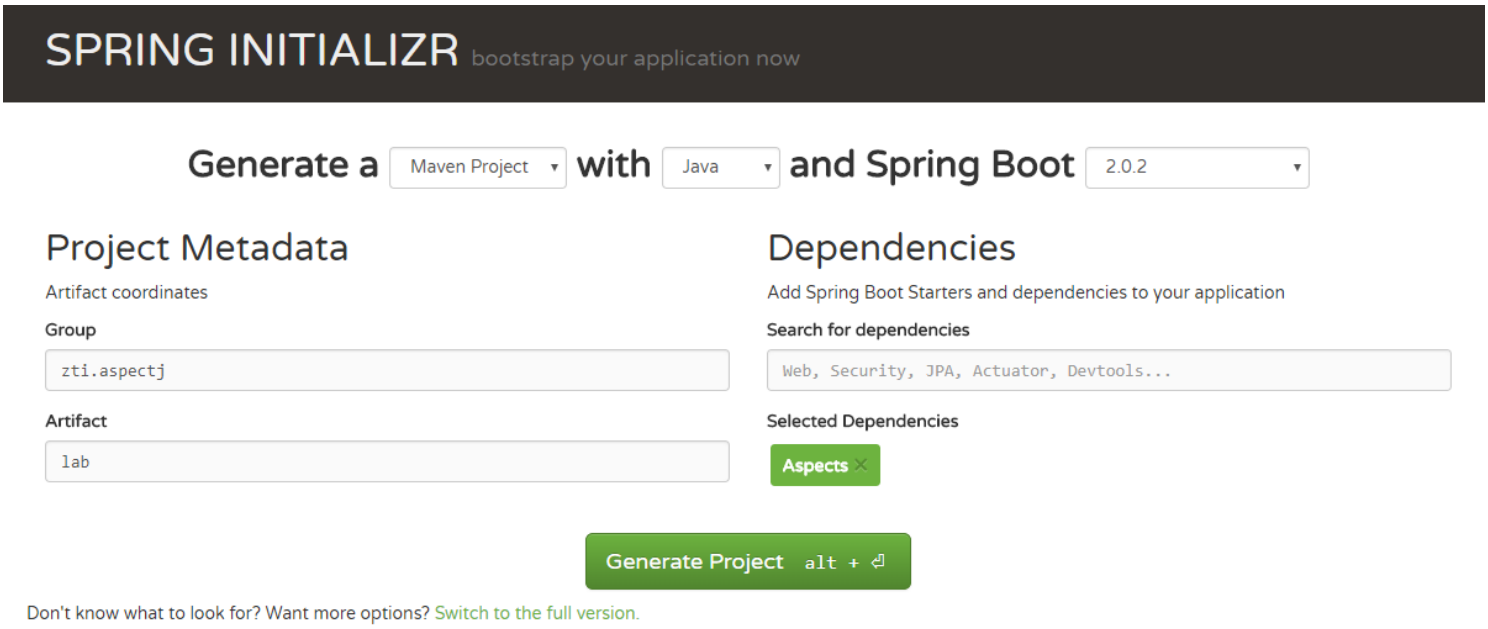
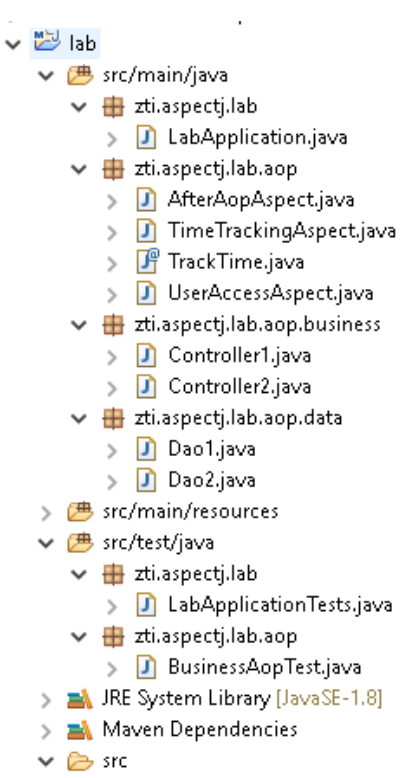
1. **Przygotowanie bootstrapowego projektu za pomocą Spring Initializr**

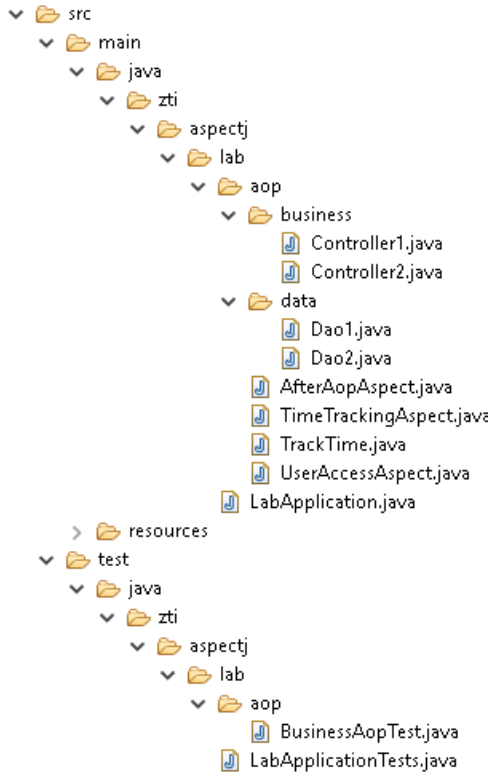
   http://start.spring.io
   Group: zti.aspectj
   Artifact: lab
   Dependencies: Aspects



2. **Import projektu Maven do środowiska Eclipse**

   Rozpakowanie pobranego katalogu
   File -> Import -> Existing Maven project
   Wskazać rozpakowany katalog i zaznaczyć plik pom.xml





3. **Utworzenie struktury folderów zgodnie z powyższym obrazkiem (prawidłowe pakiety zostaną wygenerowane podczas tworzenia folderów)**

4. **Utworzenie klas „udających" kontrolery oraz dostęp do bazy danych**
    a. Src/main/java:
    b. Pakiet zti.aspectj.lab.aop.business: https://github.com/cymonello/ZTI_lab/tree/master/zti/aspectj/lab/aop/business
        i. Controller1.java
        ii. Controller2.java
    c. Pakiet zti.apsectj.lab.aop.data: https://github.com/cymonello/ZTI_lab/tree/master/zti/aspectj/lab/aop/data
        i. Data1.java
        ii. Data2.java

    Wykorzystywane są adnotacje – "wskazówki" dla springa, dzięki którym wstrzykiwane są odpowiednie zależności
    @Service – wskazuje na klasę w warstwie logiki biznesowej
    @Repository – wskazuje na warstwę dostępu do danych
    @Autowired – służy do wstrzykiwania obiektów

    Obie klasy kontrolerów posiadają metodę CalculateSomething() -> imitacja logiki biznesowej kontrolera

5. **Stworzenie Unit Testu do sprawdzania działania aplikacji**
    a. Src/test/java:
    b. Pakiet zti.aspectj.lab.aop

i. BusinessAopTest.java
https://github.com/cymonello/ZTI_lab/blob/master/test/java/zti/aspectj/lab/aop/BusinessAopTest.java

Test realizowany za pomocą narzędzia JUnit

@RunWith(SpringRunner.class) @SpringBootTest – do celów testowych uruchamiana jest aplikacja Spring Boot
@Autowire – wstrzykiwanie klas logiki biznesowej
@Test public void invokeAOPStuff() – wywołanie metod kontrolerów

Uruchomienie testu poprzez kliknięcie na klasę prawym przyciskiem myszy i wybór Run As -> JUnitTest.

Jako że nie zostały jeszcze zaimplementowane funkcjonalności AOP, wynikiem działania aplikacji będą jedynie logi pochodzące z metod kontrolera i DAO.

6. **Implementacja AOP**
   a. Pakiet zti.aspectj.lab.aop
   https://github.com/cymonello/ZTI_lab/tree/master/zti/aspectj/lab/aop

      i. UserAccessAspect
      Aspekt wykorzystujący radę typu BEFORE – wykonanie aspektu przed metodą z punktu przecięcia
      Punkt przecięcia – wszystkie metody z pakietu lab.aop.data
      Sprawdzenie poprawności w logach

      ii. AfterAopAspect
      Rada AFTER – punkt przecięcia gdy metoda wykona się pomyślnie lub wyrzuci wyjątek
      Rada AFTERRETURNING – tylko gdy wykona się pomyślnie
      Sprawdzenie poprawności w logach

      iii. TimeTrackingAspect
      Rada @Around – przechwytuje wykonanie metody
      Utworzenie interfejsu TrackTime
      Retention policy: Runtime – adnotacja będzie dostępna podczas całego czasu działania aplikacji
      Adnotacja @Target: method – może być używana tylko do metod
      Odkomentowanie kodu w klasie Controller1
      Sprawdzenie poprawności w logach