01, 02 - WPROWADZENIE, PIERWSZY PROJEKT

Spring framework - jest to taka biblioteczka, która ułatwia w pewien sposób tworzenie złożonego oprogramowania.

Tworzenie oprogramowania na większą skalę, często to oprogramowanie ma złożoną strukturę. Składa się z wielu komponentów, którymi wraz ze wzrostem ich ilości, coraz ciężej jest zarządzać.

Spring implementuje taki wzorzec jak *Dependency Injection (Inversion of Control)* i w automatyczny sposób wczytuje sobie komponenty, sprawdza który komponent od jakiś innych zależy i ustawia im referencje między sobą.

Spring pozwala również na coś takiego jak *Programowanie Aspektowe*. W programowaniu aspektowym chodzi o to, że jak mam jakieś komponenty w całej aplikacji to często w różnych komponentach pasowałoby robić w kółko to samo. Np. tworząc aplikację bankową, ona składa się z wielu komponentów. Jeden komponent będzie do zapisywania/odczytywania klientów, inny komponent będzie do wyliczania taryfikacji. Te komponenty mają w sobie metody i chcemy w łatwy sposób (zautomatyzowany) sprawić żeby wszystkie wywołania metod w ramach tych komponentów automatycznie logowały swoje wywołania do pliku + parametry które zostały tam przekazane.

Spring również dostarcza wiele klas, które ułatwiają nam codzienne czynności np. jeśli w swojej aplikacji często korzystam z JDBC to zmagam się często z tymi samymi czynnościami. Pobieranie danych i przetwarzanie ich na obiekty biznesowe jest zadaniem nudnym, nurzącym i podatnym na błędy. Dlatego Spring dostarcza czegoś takiego co nazywa się *JDBC Template*. Jest to klasa, która automatyzuje takie codzienne czynności i pozwala na tłumaczenie wyjątków. Jeśli coś się w bazie danych posypie to odgadnięcie co się stało jest zadaniem nurzącym. Spring zamieni dziwnego exceptiona na takiego, z którym będziemy wiedzieli co zrobić.

Dodatkowo w ramach Springa, znajduje się w nim wiele podprojektów, które są specyficzne dla pewnych zastosowań. Jest np projekt, który ułatwia korzystanie z JPA czy z jakiś sieci społecznościowych.

Bardzo ważna rzecz to plik *pom.xml*, w którym będziemy musieli dodać zalezności naszego projektu od Springa (**https://mvnrepository.com/ ---> org.springframework).**

Będziemy potrzebowali:

* spring-beans
* spring-core
* spring-aop
* spring-test
* spring-context

Ale żeby tych wszystkich rzeczy nie zasysać, możemy wykorzystać fakt, że *spring-context* potrzebuje takich rzeczy jak *spring-core, spring-beans, spring-aop*. Można sprawdzić to w tabelce "depends on".

Maven ściąga wszystkie potrzebne biblioteki do *.m2/repository…*

Napiszmy sobie projekt, w którym będziemy mieli:

* Repozytorium użytkowników
  + będzie udawało, że tworzy użytkowników
  + będzie komponentem
  + będzie oparte o inny komponent (drugim komponentem będzie komponent logowania)

Stworzymy interfejs, który będzie przedstawiał operacje możliwe do wykonania w Repozytorium *użytkowników.* Potem tworzymy drugi interfejs, który będzie odpowiedzialny za logowanie.

UsersRepositorybędzie zależało od komponentu logowania.

**public** **interface** UsersRepository {

User createUser(**final** String name);

**void** setLogger(**final** Logger logger);

}

**public** **interface** Logger {

**void** log(**final** String message);

}

**public** **class** LoggerImpl **implements** Logger {

**public** **void** log(String message) {

System.***out***.println(**new** Date() + ": " + message);

}

}

**public** **class** UsersRepositoryImpl **implements** UsersRepository {

**private** Logger logger;

**public** User createUser(String name) {

logger.log("Tworzenie uzytkownika " + name);

**return** **new** User(name);

}

**public** **void** setLogger(Logger logger) {

**this**.logger = logger;

}

}

**public** **class** User {

**private** String name;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** User(String name) {

**this**.name = name;

}

}

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Logger logger = **new** LoggerImpl();

UsersRepository repository = **new** UsersRepositoryImpl();

repository.setLogger(logger);

User user = repository.createUser("Patryk");

}

}

Teraz przygotujemy sobie taki sam program tylko z użyciem - Springa.  
Spring do działania potrzebuje konfiguracji. Może się ona odbywać w Javie albo w XML'u.   
**src/main/resources** --> **Spring Bean Configuration File.**Należy wybrać przestrzenie nazw (tagi w pliku xml) - *schema/beans*. Takie taki zapisane są w pliku XSD.

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

</beans>

Pierwsze co chcemy zrobić to powiedzieć Springowi, że w naszym programie będzie się zawierał komponent, który będzie klasy *LoggerImpl.* (komponent -> bean).  
Każdy "bean" musi mieć ***identyfikator*** !!!

<bean id=*"mojLogger"* class=*"com.jurik99.springtutorial.implementation.LoggerImpl"*></bean>  
<bean id=*"repozytoriumUzytkownikow"* class=*"com.jurik99.springtutorial.implementation.UsersRepositoryImpl"*></bean>

Stosujemy implementację, nie interfejsy.

To co zostało zrobione w XML'u jest odpowiednikiem np. ***Logger logger = new LoggerImpl();***

Dalej chcemy wrzucić do implementacji RepozytoriumUzytkownika referencję do Loggera.

<bean id=*"repozytoriumUzytkownikow"* class=*"com.jurik99.springtutorial.implementation.UsersRepositoryImpl"*>

<property name=*"logger"* ref=*"mojLogger"*></property>

</bean>

To co wyżej zrobiliśmy jest odpowiednikiem tej linijki:

usersRepository.setLogget(logger);

Następnie musimy wczytać tą konfigurację. Należy uruchomić *kontekst aplikacji*. Kontekst wczytuje, mieli wszystkie ustawienia i można sobie z niego pobrać jakiegoś beana po jego nazwie.

**public** **class** MainSpring {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("konfiguracja.xml");

UsersRepository usersRepository = context.getBean("repozytoriumUzytkownikow", UsersRepository.**class**);

User user = usersRepository.createUser("Patryk");

}

}

wrz 28, 2016 1:14:36 PM org.springframework.context.support.ClassPathXmlApplicationContext prepareRefresh

INFO: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@4d405ef7: startup date [Wed Sep 28 13:14:36 CEST 2016]; root of context hierarchy

wrz 28, 2016 1:14:36 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions

INFO: Loading XML bean definitions from class path resource [konfiguracja.xml]

**Wed Sep 28 13:14:37 CEST 2016: Tworzenie uzytkownika Patryk**

*W pliku XML pokazujemy zależności pomiędzy komponentami no i zlecamy Springowi, żeby zajął się dojściem do tego w jaki sposób wstrzyknąć w różne miejsca potrzebne komponenty.*

03 - WSTRZYKIWANIE PROSTYCH ZALEŻNOŚCI

Wstrzykiwanie typów prostych. Chcemy wstrzyknąć Stringa do naszego loggera.

**public** **interface** Logger {

**void** log(**final** String message);

**void** setName(**final** String name);

}

**public** **class** LoggerImpl **implements** Logger {

**private** String name;

**public** **void** log(String message) {

System.***out***.println(**new** Date() + " [" + name + "] " + ": " + message);

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

**Wed Sep 28 15:45:28 CEST 2016 [null] : Tworzenie uzytkownika Patryk**

Wstawiło nam sie *null*, ponieważ zmienna "name" nie została nigdzie ustawiona. Żeby ją ustawić trzeba wejść do konfiguracji.xml.

<bean id=*"mojLogger"* class=*"com.jurik99.springtutorial.implementation.LoggerImpl"*>

<property name=*"name"* value=*"Mój Logger"*></property>

</bean>

<bean id=*"repozytoriumUzytkownikow"* class=*"com.jurik99.springtutorial.implementation.UsersRepositoryImpl"*>

<property name=*"logger"* ref=*"mojLogger"*></property>

</bean>

**Wed Sep 28 15:48:44 CEST 2016 [Mój Logger] : Tworzenie uzytkownika Patryk**

W każdym beanie może być znacznie więcej właściwości.

<bean id=*"mojLogger"* class=*"com.jurik99.springtutorial.implementation.LoggerImpl"*>

<property name=*"name"* value=*"Mój Logger"*></property>

<property name=*"version"* value=*"3"*></property>

</bean>

**Wed Sep 28 15:53:46 CEST 2016 [Mój Logger], version = (3) - Tworzenie uzytkownika Patryk**

Tagi <property> ***<cos tu jeszcze może być/>*** </property> mogą mieć w środku jeszcze pewne wartości, ponieważ oprócz referencji do innych beanów "ref" czy prostych wartości "value" możemy wstrzykiwać *listy, mapy, zbiory i wiele więcej…*

04 - WSTRZYKIWANIE DO KONSTRUKTORA

Problem z projektem w kroku 3. polega na tym, że dopuścilismy się zastosowania pewnego *antywzorca.*

W implementacji *UsersRepositoryImpl* zdefiniowaliśmy pole *Logger logger* i zmuszeni jesteśmy ustawić to pole korzystając z *setLogger(Logger logger).* Jednak ktoś tego może nie wiedzieć i w konfiguracji nie zdefiniuje tagu <property/>.

Możemy zdefiniowac konstruktora w *UsersRepositoryImpl*, który za parametr przyjmie beana i dzięki temu unikniemy konieczności ustawienia go setterem.

**public** **interface** UsersRepository {

User createUser(**final** String name);

}

**public** **class** UsersRepositoryImpl **implements** UsersRepository {

**private** Logger logger;

**public** UsersRepositoryImpl(Logger logger) {

**this**.logger = logger;

}

**public** User createUser(String name) {

logger.log("Tworzenie uzytkownika " + name);

**return** **new** User(name);

}

}

Podczas tworzenia beanów, Spring zagląda sobie do konfiguracji.xml i widzi, że jakiś bean jest jakieś konkretnej klasy. Następnie wywołuje *domyślny konstruktor* i tworzy obiekt danej klasy  *z użyciem refleksji.*

Dopiero po stworzeniu obiektu, wstrzykiwane są zależności.

Tworząc konstruktor z parametrem, *wyłączyliśmy konstruktor domyślny.* W pliku konfiguracyjnym korzystamy z czegoś takiego:

<bean id=*"repozytoriumUzytkownikow"* class=*"com.jurik99.springtutorial.implementation.UsersRepositoryImpl"*>

<constructor-arg ref=*"mojLogger"*></constructor-arg>

</bean>

Dodajmy jeszcze jeden parametr do konstruktora:

**public** UsersRepositoryImpl(Logger logger, String localisation) {

**this**.logger = logger;

logger.log("Lokalizacja repozytorium: " + localisation);

}

W pliku konfiguracyjnym również musimy podać wartość tego parametru:

<bean id=*"repozytoriumUzytkownikow"* class=*"com.jurik99.springtutorial.implementation.UsersRepositoryImpl"*>

<constructor-arg ref=*"mojLogger"*></constructor-arg>

<constructor-arg value=*"http://168.192.0.100:8888"*></constructor-arg>

</bean>

**Wed Sep 28 16:37:25 CEST 2016 [Mój Logger], version = (3) - Lokalizacja repozytorium: http://168.192.0.100:8888**

**Wed Sep 28 16:37:25 CEST 2016 [Mój Logger], version = (3) - Tworzenie uzytkownika Patryk**

<constructor-arg ref=*"mojLogger"*></constructor-arg> oprócz "ref" i "value" znajduje sie tutaj również kilka innych rzeczy:

* index
* name
* type

Bo może się zdarzyć, że w konfiguracji podamy parametry nie w takiej samej kolejności jak to jest zdefiniowane w konstruktorze.

**public** UsersRepositoryImpl(Logger logger, String localisation, String dbName) {

**this**.logger = logger;

logger.log("Lokalizacja repozytorium: " + localisation + "/" + dbName);

}

<bean id=*"repozytoriumUzytkownikow"* class=*"com.jurik99.springtutorial.implementation.UsersRepositoryImpl"*>

<constructor-arg ref=*"mojLogger"*></constructor-arg>

<constructor-arg value=*"databaseName"*></constructor-arg>

<constructor-arg value=*"http://168.192.0.100:8888"*></constructor-arg>

</bean>

I otrzymaliśmy niepoprawny output:

Wed Sep 28 16:51:23 CEST 2016 [Mój Logger], version = (3) - Lokalizacja repozytorium: **databaseName/http://168.192.0.100:8888**

Wed Sep 28 16:51:23 CEST 2016 [Mój Logger], version = (3) - Tworzenie uzytkownika Patryk

Domyślnie Spring stara się dopasować te argumenty w takiej kolejności w jakiej zostały podane.

Z użyciem atrybuty "name" można podać nazwy argumentów, które są podane w konstruktorze.

<constructor-arg name=*"dbName"* value=*"databaseName"*></constructor-arg>

Nie zawsze to zadziała. Kiedy?

Nie zadziała to w sytuacji kiedy spróbujemy skompilować nasz kod *bez ustawionej flagi "debug".* W takiej sytuacji VM Javy po wczytaniu naszej klasy nie będzie widziała nazw parametrów w konstruktorze.

W takiej sytuacji musimy zastosować atrybut "index".

<constructor-arg index=*"2"* value=*"databaseName"*></constructor-arg>

<constructor-arg index=*"1"* value=*"http://168.192.0.100:8888"*></constructor-arg>

Czyli numery argumentów liczone od 0, 1, 2…

Możemy również w jasny sposób powiedzieć Springowi, że parametr w konstruktorze jest typu danej klasy:

<constructor-arg type=*"java.lang.String"* value=*"50"*></constructor-arg>

05 - TWORZENIE BEANÓW Z UŻYCIEM STATYCZNYCH METOD FABRYKUJĄCYCH

Będziemy tworzyć kalendarz z ładnie sformatowaną datą.

Żeby sformatować datę potrzebujemy implementacji *DateFormat*.

**public** **static** **void** main(String[] args) {

Locale locale = Locale.*getDefault*();

Calendar calendar = GregorianCalendar.*getInstance*(locale);

DateFormat formatter = SimpleDateFormat.*getInstance*();

System.***out***.println(formatter.format(calendar.getTime()));

}

Teraz będziemy chcieli uzyskać coś takiego ale z użyciem Springa.

Najpierw stworzymy sobie beana "Locale" z użyciem metody fabrykującej *getDefault();* jednak musimy pamiętać o podaniu dodatkowego atrybutu *"factory-method".* Zostanie wywołana metoda statyczna z podanej w atrybucie "class" klasy.

<bean id=*"locale"* class=*"java.util.Locale"* factory-method=*"getDefault"*/>

<bean id=*"calendar"* class=*"java.util.GregorianCalendar"* factory-method=*"getInstance"*>

<constructor-arg ref=*"locale"*></constructor-arg>

</bean>

<bean id=*"formatter"* class=*"java.text.SimpleDateFormat"* factory-method=*"getInstance"*/>

ApplicationContext context = **new** ClassPathXmlApplicationContext("context.xml");

GregorianCalendar calendar = context.getBean("calendar", GregorianCalendar.**class**);

SimpleDateFormat formatter = context.getBean("formatter", SimpleDateFormat.**class**);

System.***out***.println(formatter.format(calendar.getTime()));

To, że w danym beanie użyty jest atrybut "class" nie znaczy to, że ten bean jest danej klasy. *Klasa ta jest wykorzystywana do tworzenia beana.*

06 - WŁASNE METODY FABRYKUJĄCE

Zajmiemy się tworzeniem beana, który przy pomocy jakieś własnej metody fabrykującej będzie tworzył obiekty.

**public** **class** User {

**private** String name;

**private** **int** age;

**private** User() {

}

**public** **static** User getInstance(**final** String name, **final** **int** age) {

User user = **new** User();

user.name = name;

user.age = age;

**return** user;

}

@Override

**public** String toString() {

**return** "User [name=" + name + ", age=" + age + "]";

}

}

<bean id=*"user"* class=*"com.jurik99.springtutorial.User"* factory-method=*"getInstance"*>

<constructor-arg type=*"java.lang.String"* value=*"Patryk"*/>

<constructor-arg value=*"25"*/>

</bean>

07 - INSTANCYJNE METODY FABRYKUJĄCE

Nie będzie tutaj metod statycznych tylko *metody pochodzące z instancji jakieś klasy.*

"someUser" będzie takiego typu jaki jest zwracany przez metodę fabrykującą "createUser()" w klasie "usersCreator".

**public** **class** User {

**private** String name;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

@Override

**public** String toString() {

**return** "User [name=" + name + "]";

}

}

**public** **class** UsersCreator {

**private** Logger logger = Logger.*getLogger*(UsersCreator.**class**.getName());

**public** User createUser(**final** String name)

{

logger.info("Tworzenie uzytkownika: " + name);

User user = **new** User();

user.setName(name);

**return** user;

}

}

<bean id=*"usersCreator"* class=*"com.jurik99.springtutorial.UsersCreator"*/>

<bean id=*"someUser"* factory-bean=*"usersCreator"* factory-method=*"createUser"*>

<constructor-arg value=*"Patryk"*/>

</bean>

Ważnym jest tutaj ***factory-bean="usersCreator"*** !!!

PRZESTRZENIE NAZW "p" i "c"

W pliku konfiguracyjnym Springa istnieją przestrzenie nazw, które ułatwiają ustawianie argumentów konstruktora jak i właściwości.

**public** **class** User {

**private** String name;

**private** **int** age;

**private** Locale locale;

**public** User(String name, **int** age, Locale locale) {

**this**.name = name;

**this**.age = age;

**this**.locale = locale;

}

}

**public** **class** SomeBean {

**private** String stringValue;

**private** **int** intValue;

**private** User user;

**public** String getStringValue() {

**return** stringValue;

}

**public** **void** setStringValue(String stringValue) {

**this**.stringValue = stringValue;

}

**public** **int** getIntValue() {

**return** intValue;

}

**public** **void** setIntValue(**int** intValue) {

**this**.intValue = intValue;

}

**public** User getUser() {

**return** user;

}

**public** **void** setUser(User user) {

**this**.user = user;

}

}

**public** **class** MainSpring {

**public** **static** **void** main(String[] args)

{

ApplicationContext context = **new** ClassPathXmlApplicationContext("context.xml");

SomeBean someBean = context.getBean("someBean", SomeBean.**class**);

System.***out***.println(someBean.getStringValue());

}

}

<bean id=*"someLocale"* class=*"java.util.Locale"* factory-method=*"getDefault"*/>

<bean id=*"someUser"* class=*"com.jurik99.springtutorial.User"*>

<constructor-arg value=*"Patryk"*/>

<constructor-arg value=*"25"*/>

<constructor-arg ref=*"someLocale"*></constructor-arg>

</bean>

<bean id=*"someBean"* class=*"com.jurik99.springtutorial.SomeBean"*>

<property name=*"user"* ref=*"someUser"*/>

<property name=*"stringValue"* value=*"some string value"*/>

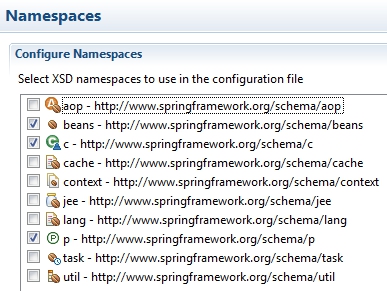
<property name=*"intValue"* value=*"100"*/>

</bean>

Ale widzimy, że niewiele zrobiliśmy a *dużo się napisaliśmy.*

Żeby to uprościć powstały *2 przestrzenie nazw (jedna się nazywa* ***"c"****, druga* ***"p"****).*

Wchodzimy do "namespaces" i zaznaczamy:



W pliku konfiguracyjnym dodały się 2 linijki:

xmlns:p=*"http://www.springframework.org/schema/p"*

xmlns:c=*"http://www.springframework.org/schema/c"*

*CONTEXT.XML Z UŻYCIEM "P" i "C":*

<bean id=*"someUser2"* class=*"com.jurik99.springtutorial.User"*

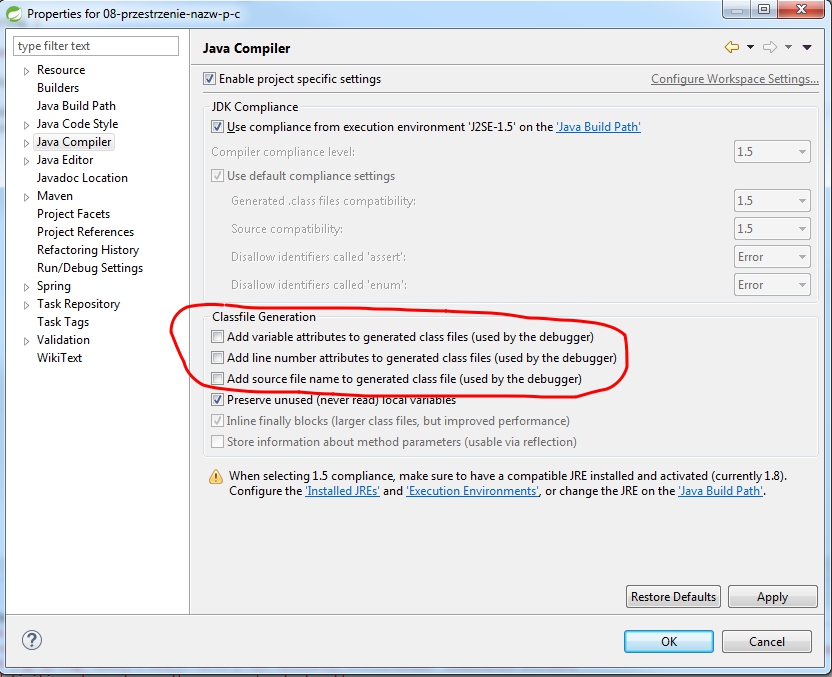
c:name=*"Patrycja"* c:age=*"25"* c:locale-ref=*"someLocale"* />

<bean id=*"someBean2"* class=*"com.jurik99.springtutorial.SomeBean"*

p:stringValue=*"some string value2"* p:intValue=*"43"* p:user-ref=*"someUser2"* />

Parametry podaje się tak samo tylko przy użyciu **"p:"**

Z tymi parametrami trzeba trochę uważać. Czasami można skompilowac kod bez informacji o debugowaniu.



Po odznaczniu tych checkboxów aplikacja się wywali

*"org.springframework.beans.factory.UnsatisfiedDependencyException"*.

Wywaliła sie dlatego, że klasy skompilowały się ale bez informacji, że kolejne parametry konstruktora np. User(String name, int age, Locale locale) mają takie a nie inne nazwy. One mogą mieć teraz nazwy np. "var1", "var2", "var3" a w konteksie odwołujemy się do "c:name", "c:age" itd… i to może być przyczyną błędów.

Kod należy kompilować z informacjami o debugowaniu. Możemy to obejść podając indeksy kolejnych parametrów:

<bean id=*"someUser3"* class=*"com.jurik99.springtutorial.User"*

c:\_0=*"Patrycja"* c:\_1=*"25"* c:\_2-ref=*"someLocale"* />

Z "p:" czyli <property/> (właściwości) nie ma takiego problemu, ponieważ ustawiamy je za pomocą setterów czyli metod a ich nazwy się nie zmieniają.

09 - TRICKS AND TIPS

http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/ Log4J

Spring pod spodem korzysta z takiej biblioteki (jest to mechanizm, który pozwala na podpięcie dowolnej implementacji logowania).

Do *Log4J* potrzebny jest plik konfiguracyjny (.properties) !!! Jego zawartość znajdziemy na stronie w linku powyżej.

log4j.rootCategory=INFO, stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %t %c**{2}**:%L - %m%n

log4j.category.org.springframework.beans.factory=DEBUG

Po ustawieniu pliku konfiguracyjnego wyświetliło nam się dużo więcej informacji:

16:59:49,740 INFO main support.ClassPathXmlApplicationContext:581 - Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@5b80350b: startup date [Mon Oct 03 16:59:49 CEST 2016]; root of context hierarchy

16:59:49,819 INFO main xml.XmlBeanDefinitionReader:317 - Loading XML bean definitions from class path resource [context.xml]

16:59:49,850 DEBUG main xml.DefaultDocumentLoader:73 - Using JAXP provider [com.sun.org.apache.xerces.internal.jaxp.DocumentBuilderFactoryImpl]

16:59:49,896 DEBUG main xml.PluggableSchemaResolver:141 - Loading schema mappings from [META-INF/spring.schemas]

16:59:49,906 DEBUG main xml.PluggableSchemaResolver:147 - Loaded schema mappings: {http://www.springframework.org/schema/cache/spring-cache-4.2.xsd=org/springframework/cache/config/spring-cache-4.2.xsd, http://www.springframework.org/schema/aop/spring-aop-4.1.xsd=org/springframework/aop/config/spring-aop-4.1.xsd, http://www.springframework.org/schema/context/spring-context-3.1.xsd=org/springframework/context/config/spring-context-3.1.xsd, http://www.springframework.org/schema/util/spring-util-3.0.xsd=org/springframework/beans/factory/xml/spring-util-3.0.xsd, http://www.springframework.org/schema/tool/spring-tool.xsd=org/springframework/beans/factory/xml/spring-tool-4.3.xsd, http://www.springframework.org/schema/aop/spring-aop-3.2.xsd=org/springframework/aop/config/spring-aop-3.2.xsd, http://www.springframework.org/schema/lang/spring-lang-4.1.xsd=org/springframework/scripting/config/spring-lang-4.1.xsd, http://www.springframework.org/schema/context/spring-context-4.0.xsd=org/springframework/context/config/spring-context-4.0.xsd, http://www.springframework.org/schema/beans/spring-beans-4.2.xsd=org/springframework/beans/factory/xml/spring-beans-4.2.xsd, http://www.springframework.org/schema/tool/spring-tool-4.1.xsd=org/springframework/beans/factory/xml/spring-tool-4.1.xsd, http://www.springframework.org/schema/lang/spring-lang-3.2.xsd=org/springframework/scripting/config/spring-lang-3.2.xsd, http://www.springframework.org/schema/cache/spring-cache-3.2.xsd=org/springframework/cache/config/spring-cache-3.2.xsd, http://www.springframework.org/schema/jee/spring-jee-4.1.xsd=org/springframework/ejb/config/spring-jee-4.1.xsd, http://www.springframework.org/schema/util/spring-util-2.0.xsd=org/springframework/beans/factory/xml/spring-util-2.0.xsd, http://www.springframework.org/schema/task/spring-task-4.2.xsd=org/springframework/scheduling/config/spring-task-4.2.xsd, http://www.springframework.org/schema/tool/spring-tool-3.2.xsd=org/springframework/beans/factory/xml/spring-tool-3.2.xsd, http://www.springframework.org/schema/context/spring-context.xsd=org/springframework/context/config/spring-context-4.3.xsd, http://www.springframework.org/schema/cache/spring-cache-4.1.xsd=org/springframework/cache/config/spring-cache-4.1.xsd, http://www.springframework.org/schema/aop/spring-aop-4.0.xsd=org/springframework/aop/config/spring-aop-4.0.xsd, http://www.springframework.org/schema/jee/spring-jee-3.2.xsd=org/springframework/ejb/config/spring-jee-3.2.xsd, http://www.springframework.org/schema/context/spring-context-3.0.xsd=org/springframework/context/config/spring-context-3.0.xsd, http://www.springframework.org/schema/util/spring-util-4.3.xsd=org/springframework/beans/factory/xml/spring-util-4.3.xsd, http://www.springframework.org/schema/util/spring-util-2.5.xsd=org/springframework/beans/factory/xml/spring-util-2.5.xsd, http://www.springframework.org/schema/beans/spring-beans-3.2.xsd=org/springframework/beans/factory/xml/spring-beans-3.2.xsd, http://www.springframework.org/schema/aop/spring-aop-3.1.xsd=org/springframework/aop/config/spring-aop-3.1.xsd, http://www.springframework.org/schema/lang/spring-lang-4.0.xsd=org/springframework/scripting/config/spring-lang-4.0.xsd, http://www.springframework.org/schema/beans/spring-beans-4.1.xsd=org/springframework/beans/factory/xml/spring-beans-4.1.xsd, http://www.springframework.org/schema/tool/spring-tool-4.0.xsd=org/springframework/beans/factory/xml/spring-tool-4.0.xsd, http://www.springframework.org/schema/lang/spring-lang-3.1.xsd=org/springframework/scripting/config/spring-lang-3.1.xsd, http://www.springframework.org/schema/cache/spring-cache-3.1.xsd=org/springframework/cache/config/spring-cache-3.1.xsd, http://www.springframework.org/schema/jee/spring-jee-4.0.xsd=org/springframework/ejb/config/spring-jee-4.0.xsd, http://www.springframework.org/schema/task/spring-task-4.1.xsd=org/springframework/scheduling/config/spring-task-4.1.xsd, http://www.springframework.org/schema/tool/spring-tool-3.1.xsd=org/springframework/beans/factory/xml/spring-tool-3.1.xsd, http://www.springframework.org/schema/cache/spring-cache-4.0.xsd=org/springframework/cache/config/spring-cache-4.0.xsd, http://www.springframework.org/schema/jee/spring-jee-3.1.xsd=org/springframework/ejb/config/spring-jee-3.1.xsd, http://www.springframework.org/schema/util/spring-util-4.2.xsd=org/springframework/beans/factory/xml/spring-util-4.2.xsd, http://www.springframework.org/schema/task/spring-task-3.2.xsd=org/springframework/scheduling/config/spring-task-3.2.xsd, http://www.springframework.org/schema/beans/spring-beans-3.1.xsd=org/springframework/beans/factory/xml/spring-beans-3.1.xsd, http://www.springframework.org/schema/util/spring-util.xsd=org/springframework/beans/factory/xml/spring-util-4.3.xsd, http://www.springframework.org/schema/aop/spring-aop-3.0.xsd=org/springframework/aop/config/spring-aop-3.0.xsd, http://www.springframework.org/schema/beans/spring-beans-4.0.xsd=org/springframework/beans/factory/xml/spring-beans-4.0.xsd, http://www.springframework.org/schema/beans/spring-beans.xsd=org/springframework/beans/factory/xml/spring-beans-4.3.xsd, http://www.springframework.org/schema/lang/spring-lang-3.0.xsd=org/springframework/scripting/config/spring-lang-3.0.xsd, http://www.springframework.org/schema/context/spring-context-2.5.xsd=org/springframework/context/config/spring-context-2.5.xsd, http://www.springframework.org/schema/context/spring-context-4.3.xsd=org/springframework/context/config/spring-context-4.3.xsd, http://www.springframework.org/schema/task/spring-task-4.0.xsd=org/springframework/scheduling/config/spring-task-4.0.xsd, http://www.springframework.org/schema/tool/spring-tool-3.0.xsd=org/springframework/beans/factory/xml/spring-tool-3.0.xsd, http://www.springframework.org/schema/aop/spring-aop-2.0.xsd=org/springframework/aop/config/spring-aop-2.0.xsd, http://www.springframework.org/schema/jee/spring-jee-3.0.xsd=org/springframework/ejb/config/spring-jee-3.0.xsd, http://www.springframework.org/schema/util/spring-util-4.1.xsd=org/springframework/beans/factory/xml/spring-util-4.1.xsd, http://www.springframework.org/schema/task/spring-task-3.1.xsd=org/springframework/scheduling/config/spring-task-3.1.xsd, http://www.springframework.org/schema/beans/spring-beans-3.0.xsd=org/springframework/beans/factory/xml/spring-beans-3.0.xsd, http://www.springframework.org/schema/jee/spring-jee.xsd=org/springframework/ejb/config/spring-jee-4.3.xsd, http://www.springframework.org/schema/aop/spring-aop-2.5.xsd=org/springframework/aop/config/spring-aop-2.5.xsd, http://www.springframework.org/schema/aop/spring-aop-4.3.xsd=org/springframework/aop/config/spring-aop-4.3.xsd, http://www.springframework.org/schema/lang/spring-lang-2.0.xsd=org/springframework/scripting/config/spring-lang-2.0.xsd, http://www.springframework.org/schema/util/spring-util-3.2.xsd=org/springframework/beans/factory/xml/spring-util-3.2.xsd, http://www.springframework.org/schema/task/spring-task.xsd=org/springframework/scheduling/config/spring-task-4.3.xsd, http://www.springframework.org/schema/tool/spring-tool-2.0.xsd=org/springframework/beans/factory/xml/spring-tool-2.0.xsd, http://www.springframework.org/schema/lang/spring-lang-2.5.xsd=org/springframework/scripting/config/spring-lang-2.5.xsd, http://www.springframework.org/schema/lang/spring-lang-4.3.xsd=org/springframework/scripting/config/spring-lang-4.3.xsd, http://www.springframework.org/schema/context/spring-context-4.2.xsd=org/springframework/context/config/spring-context-4.2.xsd, http://www.springframework.org/schema/jee/spring-jee-2.0.xsd=org/springframework/ejb/config/spring-jee-2.0.xsd, http://www.springframework.org/schema/tool/spring-tool-4.3.xsd=org/springframework/beans/factory/xml/spring-tool-4.3.xsd, http://www.springframework.org/schema/tool/spring-tool-2.5.xsd=org/springframework/beans/factory/xml/spring-tool-2.5.xsd, http://www.springframework.org/schema/jee/spring-jee-4.3.xsd=org/springframework/ejb/config/spring-jee-4.3.xsd, http://www.springframework.org/schema/jee/spring-jee-2.5.xsd=org/springframework/ejb/config/spring-jee-2.5.xsd, http://www.springframework.org/schema/util/spring-util-4.0.xsd=org/springframework/beans/factory/xml/spring-util-4.0.xsd, http://www.springframework.org/schema/task/spring-task-3.0.xsd=org/springframework/scheduling/config/spring-task-3.0.xsd, http://www.springframework.org/schema/cache/spring-cache-4.3.xsd=org/springframework/cache/config/spring-cache-4.3.xsd, http://www.springframework.org/schema/aop/spring-aop-4.2.xsd=org/springframework/aop/config/spring-aop-4.2.xsd, http://www.springframework.org/schema/lang/spring-lang.xsd=org/springframework/scripting/config/spring-lang-4.3.xsd, http://www.springframework.org/schema/context/spring-context-3.2.xsd=org/springframework/context/config/spring-context-3.2.xsd, http://www.springframework.org/schema/util/spring-util-3.1.xsd=org/springframework/beans/factory/xml/spring-util-3.1.xsd, http://www.springframework.org/schema/beans/spring-beans-2.0.xsd=org/springframework/beans/factory/xml/spring-beans-2.0.xsd, http://www.springframework.org/schema/cache/spring-cache.xsd=org/springframework/cache/config/spring-cache-4.3.xsd, http://www.springframework.org/schema/lang/spring-lang-4.2.xsd=org/springframework/scripting/config/spring-lang-4.2.xsd, http://www.springframework.org/schema/context/spring-context-4.1.xsd=org/springframework/context/config/spring-context-4.1.xsd, http://www.springframework.org/schema/beans/spring-beans-2.5.xsd=org/springframework/beans/factory/xml/spring-beans-2.5.xsd, http://www.springframework.org/schema/beans/spring-beans-4.3.xsd=org/springframework/beans/factory/xml/spring-beans-4.3.xsd, http://www.springframework.org/schema/tool/spring-tool-4.2.xsd=org/springframework/beans/factory/xml/spring-tool-4.2.xsd, http://www.springframework.org/schema/jee/spring-jee-4.2.xsd=org/springframework/ejb/config/spring-jee-4.2.xsd, http://www.springframework.org/schema/task/spring-task-4.3.xsd=org/springframework/scheduling/config/spring-task-4.3.xsd, http://www.springframework.org/schema/aop/spring-aop.xsd=org/springframework/aop/config/spring-aop-4.3.xsd}

16:59:49,915 DEBUG main xml.PluggableSchemaResolver:119 - Found XML schema [http://www.springframework.org/schema/beans/spring-beans.xsd] in classpath: org/springframework/beans/factory/xml/spring-beans-4.3.xsd

16:59:50,005 DEBUG main xml.DefaultBeanDefinitionDocumentReader:92 - Loading bean definitions

16:59:50,029 DEBUG main xml.XmlBeanDefinitionReader:224 - Loaded 1 bean definitions from location pattern [context.xml]

16:59:50,072 DEBUG main support.DefaultListableBeanFactory:720 - Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@28d25987: defining beans [someString]; root of factory hierarchy

16:59:50,073 DEBUG main support.DefaultListableBeanFactory:221 - Creating shared instance of singleton bean 'someString'

16:59:50,074 DEBUG main support.DefaultListableBeanFactory:448 - Creating instance of bean 'someString'

16:59:50,103 DEBUG main support.DefaultListableBeanFactory:529 - Eagerly caching bean 'someString' to allow for resolving potential circular references

16:59:50,106 DEBUG main support.DefaultListableBeanFactory:484 - Finished creating instance of bean 'someString'

16:59:50,108 DEBUG main support.DefaultListableBeanFactory:251 - Returning cached instance of singleton bean 'lifecycleProcessor'

16:59:50,124 DEBUG main support.DefaultListableBeanFactory:251 - Returning cached instance of singleton bean 'someString'

Dostępne jest również coś takiego jak *źródła Springa, żeby sprawdzić co tak naprawdę w środku tego Springa się dzieje.*

PPM na projekcie / Maven / Download sources i teraz trzymając CTRL możemy wejść do źródła np:

**"ClassPathXmlApplicationContext"** i przejrzeć jak ona działa, debugować itd…

10 - BEANY WEWNĘTRZNE I WSTRZYKIWANIE NULLI

WAŻNE POJĘCIA I KOMENTARZE

**Difference between <context:annotation-config> VS <context:component-scan>**

<context:annotation-config> is used to activate annotations in beans already registered in the application context (no matter if they were defined with XML or by package scanning).

<context:component-scan> can also do what <context:annotation-config> does but <context:component-scan> also scans packages to find and register within the application context.