Class 3 – Spring Boot Microservice Enterprise Applications with Java

Contents

Review

- Class2
- Q&A

Break

Spring Framework Overview[1]

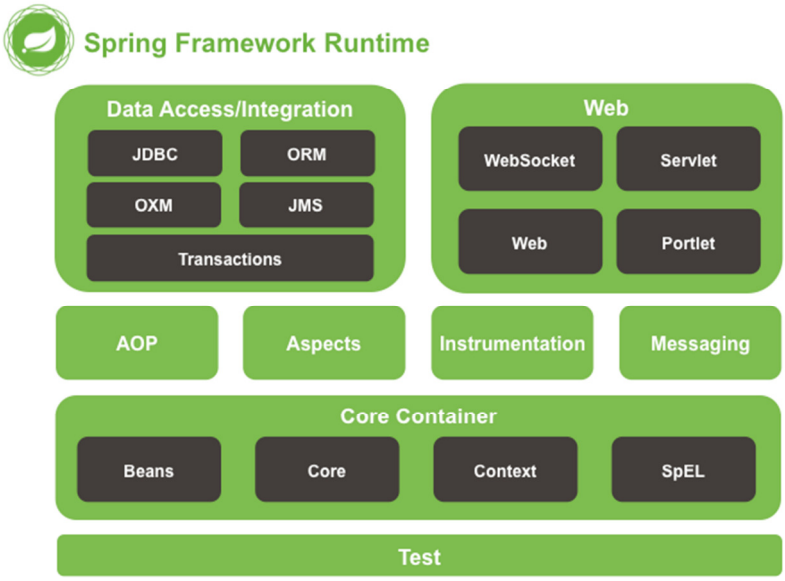
"The Spring Framework is a lightweight solution and a potential one-stop-shop for building your enterprise-ready applications."

[1] http://docs.spring.io/spring/docs/current/spring-framework-reference/html/overview.html

History

Spring started out as an inversion of control container to simplify the complexities of enterprise Java development and specially EJB's. Spring does not want to be in your code (non-intrusive) or force you to implement certain interfaces. It started in 2004 so it is stable and mature.

Overtime it has grown to include a number features and projects. It is broken into multiple modules that you may pick and choose. We will use a few.



- Spring is built around design patterns (gang of four[2]) and best practices (**all beans are by default Singletons**) and as such has the programmer learn and use them even without knowing
- Improves testability as it decouples beans and allows for much easier unit testing with mocks
- Spring Context can be thought of as registry (the ApplicationContext) of Java object instances and their web of interdependencies. Spring instantiates all the necessary objects and their dependencies in necessary order (make sure you do not define cyclical dependencies!)
- It is an integration of many different technologies (above diagram)

[2] http://en.wikipedia.org/wiki/Design_Patterns

POJOs and Beans

POJO = Plain Old Java Object

Bean = a Java object with getters and setter (a bit unfortunate as it goes against encapsulation) so think twice before creating getter/setters for everything. Every attribute DOES NOT require getter/setters.

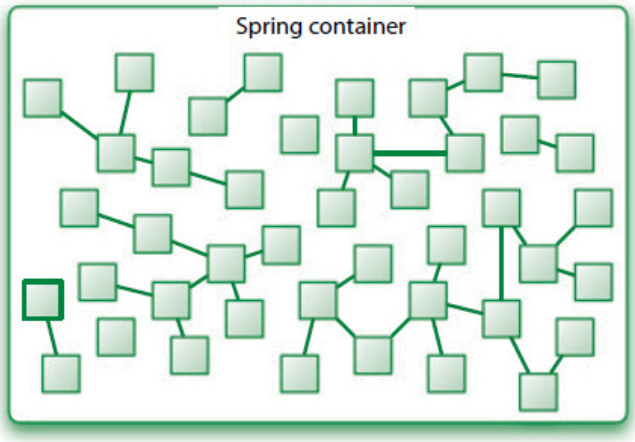


Figure 1.4 In a Spring application, objects are created, wired together, and live within the Spring container.

JUnit and Spring

JUnit is a very popular unit testing framework (TestNG is another one) and Spring provides an excellent integration for both.

Wiring Beans using XML

Hands On - Using Example from here: https://github.com/rahulaga/hello-spring

Try it yourself as well (recall **git clone**)

Break

Wiring Beans using Annotations and Code

Hands On - Using Example from here: https://github.com/rahulaga/hello-spring-annotation

Try it yourself as well.

XML configuration vs Annotations

- XML is verbose but nice to see the application context in one place
- XML configuration can be split into multiple files
- Annotations in code so excellent context but they "bring Spring into your code" and no single view
- A third way is to actually implement Spring interfaces – not recommended
- It is good to know the various methods as you may work at a company where a different method is used than what you are used to

- If it is not your source code then you cannot add annotations and must use XML or Programmatic bean creation

For rest of my examples I will be using Annotations and code. This is the Spring Boot convention as well

JSR-330[3] Annotations

The standard Java annotations are slightly different from Spring's which we have used so far. Spring supports JSR-330 as well. You can use these if you prefer.

Spring	JSR-330
@Autowired	@Inject
@Component	@Named

There are some others as well and there are some subtle differences. You can read online if you wish[4]. We will come across more in later classes.

More on Beans

- Scope[5] – singleton (default) and prototype
  - <bean scope="prototype"… />
  - @Scope
- Initialization (init-method) and Destruction (destroy-method)
  - <bean init-method="start" destroy-method="stop"…. />
  - @PostConstruct
  - @PreDestroy
- Wiring collections - http://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/htmlsingle/#beans-collection-elements

Bean Lifecycle[6]

1. Create bean instance from the definition
2. Populate properties (Constructor Args, Setters)
3. PreInitilization (BeanPostProcessor)
4. @PostConstruct
5. PostInitilization (BeanPostProcessor)
6. Bean Ready to use
7. @PreDestroy

[3] https://jcp.org/aboutJava/communityprocess/final/jsr330/
[4] http://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/htmlsingle/#beans-standard-annotations
[5] There are more scopes: http://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/htmlsingle/#beans-factory-scopes
[6] https://docs.spring.io/spring/docs/current/spring-framework-reference/html/beans.html

Homework for Class 3

Try out Spring examples.