Keywords Summary

This document is only a summary of keyword basic explanation regarding Spring. It does not include detailed implementations.

Spring IoC

1. IoC

Inversion of Control is a design principle that allows classes to be loosely coupled and therefore easier to test and maintain. IoC refers to transferring the control of objects and their dependencies from the main program to a container or framework. It is a principle, not a design pattern.
3 implementations:

- Dependency injection
- Factory design pattern
- Strategy design pattern

2. DI

Dependency injection is a technique that allows objects to be separated from the objects they depend upon. There are 3 approaches:

- Constructor-based:
- Setter-based
- Field-based

3. Bean

In Spring, the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container.

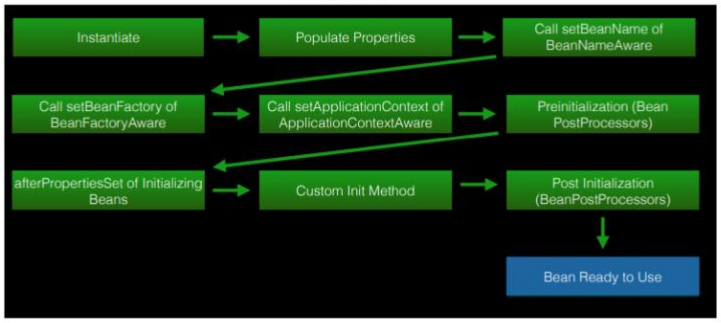
4. Bean Scope

- Singleton: Scopes a single bean definition to a single object instance per Spring IoC container. (default)
- Prototype: Scopes a single bean definition to any number of object instances.
- Request: Scopes a single bean definition to the lifecycle of a single HTTP request; that is each and every HTTP request will have its own instance of a bean created off the back of a single bean definition. Only valid in the context of a web-aware Spring *ApplicationContext*.
- Session: Scopes a single bean definition to the lifecycle of a HTTP Session. Only valid in the context of a web-aware Spring *ApplicationContext*.
- Global session: Scopes a single bean definition to the lifecycle of a global HTTP Session. Typically only valid when used in a portlet context. Only valid in the context of a web-aware Spring *ApplicationContext*.

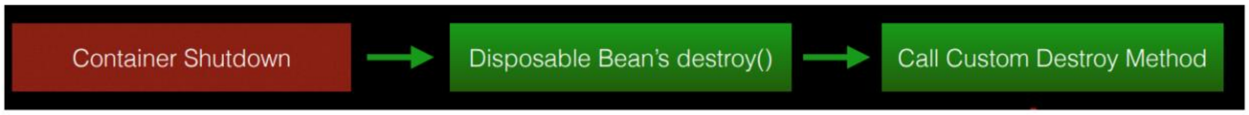
5. Bean Life Cycle

General stages:Container Started -> Bean Instantiated -> Dependencies Injected -> Custom init method -> Custom utility method -> Custom destroy method
Part 1: Show the stages a bean goes through after instantiation until it is ready for use.
Part 2: Shows what happens to a bean once the Spring IoC container shut downs

Part 1


Instantiate
Populate Properties
Call setBeanName of BeanNameAware
Call setBeanFactory of BeanFactoryAware
Call setApplicationContext of ApplicationContextAware
Preinitialization (Bean PostProcessors)
afterPropertiesSet of Initializing Beans
Custom Init Method
Post Initialization (BeanPostProcessors)
Bean Ready to Use

Part 2



Spring AOP

1. AOP

Aspect-oriented programming uses aspects in programming. It breaks code into different modules where the aspect is the key unit of modularity. Aspects enable the implementation of crosscutting concerns such as transactions, logging, etc.

2. Aspect

An aspect is a modularization of a concern that cuts across multiple classes, such as transactions, logging, security.
The annotation is @Aspect

3. Advice

An advice is an action taken by an aspect at a particular Joinpoint. Different types of advice include:

- Before: run advice before the method execution
- After: run advice after the method execution, regardless of its outcome
- After-returning: after execution, only if the method completes successfully
- After-throwing: after execution, only if the method exits by throwing an exception
- Around: run advice before and after the advised method is invoked.
-

4. JoinPoint
   It is a point during the execution of a program, such as the execution of a method or the handling of an exception. In Spring AOP, a JoinPoint always represents a method execution.
5. PointCut
   It is a predicate that helps match an Advice to be applied by an Aspect at a particular JoinPoint. We often associate the Advice with a Pointcut expression, and it runs at any Joinpoint matched by the Pointcut.
6. Target Object
   A target object is an object being advised by one or more aspects. In Spring AOP, a subclass is created at runtime where the target method is overridden and advices are included based on their configuration.
7. Proxy
   It is an object that is created after applying advice to the target object.
8. @Transactional
   This annotation describes a transaction attribute on an individual method or on a class.

Spring MVC

1. MVC: Model, View, Controller
   Spring MVC is a Java framework used to build web applications. It follows the Model-View-Controller design pattern. It implements all the basic features of a core spring framework like IoC, DI.
   A Spring MVC provides an elegant solution to use MVC in spring framework by the help of DispatcherServlet.
2. Spring MVC workflow
   The incoming request is intercepted by the DispatcherServlet that works as the front controller. Then the DipatcherServlet gets an entry of handler mapping and forwards the request to the Controller and the Controller returns an object of ModelAndView. After that, the DispatcherServlet checks the entry of ViewResolver and invokes the specified View component.

Spring Boot

Spring Boot is an open-source Java-based framework used to create a micro Service. It provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can just run. It enables you to get started with minimum configurations without the need for an entire Spring configuration setup.

1. Spring Boot advantages
   - Easy to understand and develop Spring applications
   - Increase productivity
   - Reduce the development time

2. Spring Boot Starter
   Starter POMs are a set of convenient dependency descriptors that you can include in your application. You get a one-stop shop for all the Spring and related technology that you need, without having to hunt through sample code and copy-paste loads of dependency descriptors.
3. Auto Configuration
   Spring Boot auto-configuration attempts to automatically configure your Spring application based on the jar dependencies that have been added.
4. Rest API Design
   REST stands for representational state transter. API is a contract between an information provider and an information user. It establishes the content required from the request and the content required by the response.
   There are 3 aspects in RESTful APIs Design:
   - Resources: URL
   - Resource representations:
     - Html
     - XML(old fashion) -> JSON(modern version)
   - HTTP methods
5. Spring Restful API
   - get /api/user/{uid}
   - get (with filter) /api/user?pageNo=2&rows=10&orderBy=salary
   - post /api/user
   - put /api//user/{id}
   - delete /api//user/{id}

Exception Handling Process

- ExceptionHandler -> local exceptions
- ControllerAdvice -> global exceptions

Validation

Use spring-boot-starter-validation to check if the data follows the format required.

Swagger

Swagger is a set of open-source tools built around the OpenAPI Specification that help you design, build, document, and consume REST APIs. It allows you to describe the structure of your APIs so that machines can read them.