BULZAN STEFAN 26.04.2020

SPRING FRAMEWORK INTRO

Spring Bean Container

FIRST WE TALK...

ABOUT INFRASTRUCTURE

- Roads, electrical grid, plumbing (water, sewage)
- What is the purpose of infrastructure?
  - Is to provide access to services that are used by everyone
- It doesn’t matter the REASON you use the infrastructure, it is delivered to you
- What if for each house you would have to build also the water pipes from the river, or build your own road again and again
- When you build a house, you just connect to the infrastructure that is provide to you by a 3rd party (the municipality)
- The infrastructure is NOT built specifically for your house, it can be used from any house
- Think about how it would be if only red cars could drive on the roads, or you could use water only for chicken soup, and for vegetable soup you need other pipes

WHAT IS A FRAMEWORK

- When you build an application there are a lot of things that are the same for every application
- There are infrastructural concerns:
  - How we expose the data to the world
  - What format we use
  - What protocol we use, and how we support that in our application
  - How we deal with the database connection
  - How we link the business objects
- All these concerns can be abstracted and implemented without knowing exactly our domain

WHAT DOES SPRING OFFER

- Spring is an umbrella term for multiple projects:
  - Spring Core : provides a basic Dependency Injection mechanism
  - Spring Web : provides a way to define controllers and provide data using HTTP protocol
  - Spring Data : provides a way to easily connect to the database, relational or non-relational
  - Spring Boot : provides a complete package to run a web application with any other project you want to enable
  - Spring Security: provides mechanisms to secure the access to your application

SPRING CORE

- The spring core provides the basic objects that are used to implement the other Spring projects
- The main feature of spring framework, upon which all other spring projects are built is the BEAN CONTAINER
- What is a bean?
  - A simple POJO (Plain Old Java Object) that is recognised by Spring and accessible through the Bean Container
- What is the bean container?
  - A collection of objects managed by Spring
  - You can define objects that will be in the container
  - You can request for objects from the container for usage

DEPENDENCY INJECTION (DI)

- Or Inversion Of Control (IOC)
- The problem is who is responsible of creating an object
- For example, I have a PersonService that uses a database
  - Who is responsible of creating the Database Access Object?
  - Who is responsible of handling the connection, connection credentials, what to do when connection fails, reconnect, etc
  - Surely PersonService should handle only the logic regarding… Persons, Connection concerns shouldn’t be in PersonService
- In this case, Dependency Injection comes to rescue: PersonService will only request the Data Access Object to be injected into its state, and it will work with the object without caring about the connection complexity

DECLARING SPRING BEANS

- To declare an object as a bean annotate it with
  - @Component
  - @Service : alias for @Component, it only shows that this bean is a service
  - @Repository: alias for @Component, it denotes the object as being a data access object
- The beans are saved in the bean container having name the name of the class starting with lower case: eg. PersonService will have the name personService
- If you want to change the name of the bean or you have multiple beans with the same type, you can use an alias for @Component(“alias”) and access it with @Qualifier(“alias”)

SPRING BEAN CONTAINER = DI IMPLEMENTATION

- The spring bean container is an implementation of Dependency Injection
- Spring uses annotation to implement features
- At startup Spring Framework will scan all classes from the current package and its sub-packages and search for classes that have @Component and the other annotations
  - If it finds an object with @Component, it instantiates it and keeps the reference in the Spring Container
  - The Spring Bean objects must
    - Have a no-arg constructor
    - Have all arguments in constructor be Spring Beans
  - This way the Spring is able to instantiate the object

SPRING BOOT: CREATING AN APPLICATION

- Spring Boot provides a simple way to start a Spring application. You can use Spring without Spring Boot, but you'll have to make much more configurations to make it work
- Go to https://start.spring.io/
- Select Maven and Java
- Select latest spring version (no snapshot/or M*)
- Insert the data requested (group, artefact, name, description,...)
- Packaging: JAR
- Java: latest you have available (14)
- Select dependencies (for spring core none is required)
- GENERATE

DEMO

CREATING A SPRING BEAN

USING BEANS

- Now that we have some beans in the container, what can we do with them?
- Well, use them:
  - In order to use spring beans, the object must be also a spring bean
- So, we have a Spring Bean that depends on another one. You can inject that object with @Autowired :
  - RECOMMENDED: in the constructor (annotate constructor with @Autowired if there are multiple constructors)
  - Setters : put @Autowired for the setters you want to inject from the container
  - Fields : put @Autowired directly on the fields. Never use it, just know it exists
-

DEMO

INJECT BEANS IN ANOTHER ONE

NOTES

- We use beans for Business Objects (Controller, Service, Repository), not for model
- When you declare a class @Component, there will be just ONE instance in the bean repository
- The model contains lots of instances of the same class, so it shouldn't be a bean
- You can also create beans by having a @Configuration class and annotating a method with @Bean and the return value will be stored in the Spring Container