Day 1

Monday, August 22, 2022 1:41 PM

Spring Intro

family of frameworks, helps us create loosely coupled java applications quickly

It is an inversion of control container this is what provides the flexible infrastructure

The main IOC is dependency Injection

Spring allows us to build applications using POJO's, and applying services to those pojos. Spring manages these services and other infrastructure needs allowing us to focus logic

The Spring modules we should know
- Core
- Web/MVC
- ORM
- AOP
- Security

Spring Core:

Contains the basic framework and IOC Container
- Core and Beans: fundamental framework IOC container
- Beanfactory: one of the IOC containers
  - Create beans, do the dependency injection
  - Bean: is a class managed by Spring
- Context: provide the other IOC container
  ApplicationContext (preferred container)
- Annotations for spring framework

Spring IOC Containers / Dependency Injection

Inversion of Control: a design pattern which control over certain parts of an application are inverted / handed to another entity to achieve loose coupling

IOC container is in charge of:
- instantiating
- configuring
- assembling the beans

We put this information in an xml file
- beans.xml
- applicationContext.xml

Two types of IOC Containers
- Beanfactory: base container
- ApplicationContext: Child of Beanfactory
  - Same features of Beanfactory plus
    - integration with AOP
    - event propagation
    - message resource handling
    - application layer specific context

Dependency Injection
- removes dependencies from a program by providing the configuration in an external file

This loosely couples our code, makes it easier test, and implement in a wider variety of environments

Two ways of achieving DI in Spring:

Constructor Injection: spring will use the objects Constructor to provide the dependency

to provide the dependency

Setter Injection: will call the no-arg constructor, the use the setter method to set the dependency

Spring Bean Lifecycle
Beans are managed by the Beanfactory or ApplicationContext and they have a lifecycle, here is the high level view

1. Beans are instantiated
2. Properties of the beans are set
3. Any associated interfaces are made aware of their existance
4. The bean is made aware of any associated interfaces
5. Some/any custom startup/creation methods invoked
6. Bean is ready to use
7. It gets marked for removal, destroy method is called
8. Any custom destroy methods called
9. Bean is destroyed

Scopes of Beans

In the case of beans scope can be seen as a subsection of a larger application, with certain, values properties and objects

We have six scopes:

Singleton: default, one single instance of that bean for the app

Prototype: single mapped to multiple object instances

Request: a bean scoped to the lifetime of a HTTP request

Session: bean scoped to the lifetime of a HTTP Session

Global Session: scoped to a global HTTP session

Application: scoped to a servlet context