Day 4 Revisit

Spring Framework – Spring is a popular Java Framework to create loosely coupled Enterprise Java Applications.

It contains many modules

* Spring Core
* Spring Data
* Spring Web/MVC
* Spring AOP
* Spring Security
* Spring Test

Spring framework heavily depends on two important design patterns

* Inversion of Control (IoC)
* Dependency Injection (DI)

Core Module contains the following packages

* Spring core jar
* Spring beans jar
* Spring context jar
* Spring Expression jar (log.jar,common-logging.jar)

IoC – Managing the Bean Life cycle (IoC container)

DI – Injecting the required/dependent objects at run time

Way of creating/configure beans using spring framework

1. Using XML configuration
2. Using Annotations or Java files

Creating Stand-alone Spring core applications

* Resource & BeanFactory (XmlBeanFactory)
* ApplicationContext (ClasspathXmlApplicationContext)

Types of Dependency Injection

* Constructor Injection (xml file <constructor-arg> value&type) default data type is String
* Setter Injection (xml file <property> name& value)

Wiring the Bean – Creating dependent objects automatically

Autowiring – no, byname, byClass, byType, Constructor (xml file autowire=””, or @Autowired)

Types of Annotations

1. Spring Core Annotations
2. Spring Boot Annotations
3. Spring Data Annotations
4. Spring Security Annotations
5. Spring Cloud Annotations
6. Spring Web Annotations

Core Annotations

1. DI Annotations (@Autowired, @Primary, @DependsOn, @Qualifier,)
2. Context Config Annotations (@Value, @PropertySource, @PropertySources, @Import, @Config)

Spring Boot Annotations

@RestController

@SpringBootApplication

@AutoConfigure

@Component-scan

Spring Data Annotation (Persistence Annotations)

@Entity

@Table

@Id

@Column

@GeneratedValue

@OneToMany

@OneToOne

@ManyToMany

@NamedQuery

@NamedQueries

@Transactions

@Repository

Spring Cloud Annotations

@EnableEurekaServer

@EnableEurekaClient

@EnableSwagger

Spring Web Annotations

@Controller

@RequestMapping

@GetMapping

@PostMapping

@PutMapping

@DeleteMapping

@RequestBody

@PathVariable

Spring MVC based Web Applications using Spring MVC & hibernate.

Crud Operations using Spring and Hibernate Framework.

Spring Framework is also called as Framework of Frameworks.

Challenges in Spring Framework

* Configuring the XML files

Spring Boot is a way of creating Spring based Enterprise applications easily and quickly.

Different ways of creating Spring boot applications

1. Using Spring Initializr (Online form) – <https://start.spring.io>
2. Using STS (Flavour of Eclipse) [Spring Tools Suite]
3. Using Spring Boot CLI (Command Line Interface – command based)

Spring Boot is mainly used to create Web Service.

Web Service – Service through Web (Internet)

Internet uses HTML lang and http protocol

Types of Web Service

* SOAP based (Simple Object Access Protocol) – XML based web Service development. It uses WSDL (WebService Definition/Description Language)
* REST based ( Representational State Transfer) – http – It Re-uses http protocol

HTTP Methods

* Get
* Post
* Put
* Delete
* Options
* Trace

Post Man is a Web Service Client (Make any type of http request)

API – Application Programming Interface

Web Service

* End points
* URI (Uniform Resource Identifier)
* URL (Uniform Resource Locator)

Base package

Controller package

Entity/Model Package

DAO/Repository package

Service package

Exception package

Utility Package

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl No** | **Http Method** | **SQL Operation** | **Method Signature** | **Return Value** | **Repo Method** |
| 1 | Get | ReadAll | getBean() | List<Bean> | findAll() |
| 2 | Get | ReadById | getBeanById(Type t) | Bean | findById(Type t) |
| 3 | Post | Insert/Add | addBean(Bean obj) | void | Save(Entity e) |
| 4 | Put | Edit/Update | editBean(Type t, Bean obj) | Void | Save(Entity e) |
| 5 | Delete | DeleteById | deleteBean(Type t) | void | deleteById(Type t) |

1. Define Entity Bean class with all required Annotations
2. Create a EntityRepository interface by extending JpaRepository<Entity, ID>
3. Create a Service interface and add all the required methods
4. Create a ServiceImpl class and provide implementation to all the service methods using Autowired EntityRepository reference
5. Create a Controller for all the CRUD end points using Autowired ServiceInterface reference.