Day 16 Spring Boot Deep Dive.

Day 15 Revisit IO, NIO & Serialization with Http Client.

IO Streams - Raw Data (img files, audio/video files – Byte Streams, each chunck is a single byte (8 bits)) Text based Data – (Character Streams)

IO Operations (Blocking Operations in Nature)

NIO (Non-Blocking Async Operation) – Channel.

Serialization & De-Serialization concepts

HttpClient –

Airline Reservation System.

CLI based application for the same. Considering few modules (User Mgmt, Profile Mgmt, Aircraft Mgmt, Reservation Mgmt, Payment Mgmt)

Web Based application using Spring Boot and MySql/SQL Server DB.

User Mgmt (Users & Roles)

Passengers, Staffs, SuperUser (Admin)

Role Based Access.

Web-Based Application – ARS (Airline Reservation System)

Spring Boot

Spring is a Popular Java based framework which helps to create loosely coupled enterprise applications.

Spring Boot – Is a simplified way of creating Spring based application

IoC – Inversion of Control

DI – Dependency Injection

Spring Uses Spring -Beans --- Managed Objects

Instead of Creating Object manually using new Keyword along with the constructor, framework will take care of creating and managing it’s life cycle.

//Normal Style Object Creation

Employee emp = new Employee(100, “test”, “[test@gmail.com](mailto:test@gmail.com)”);

Spring Style Object Creation

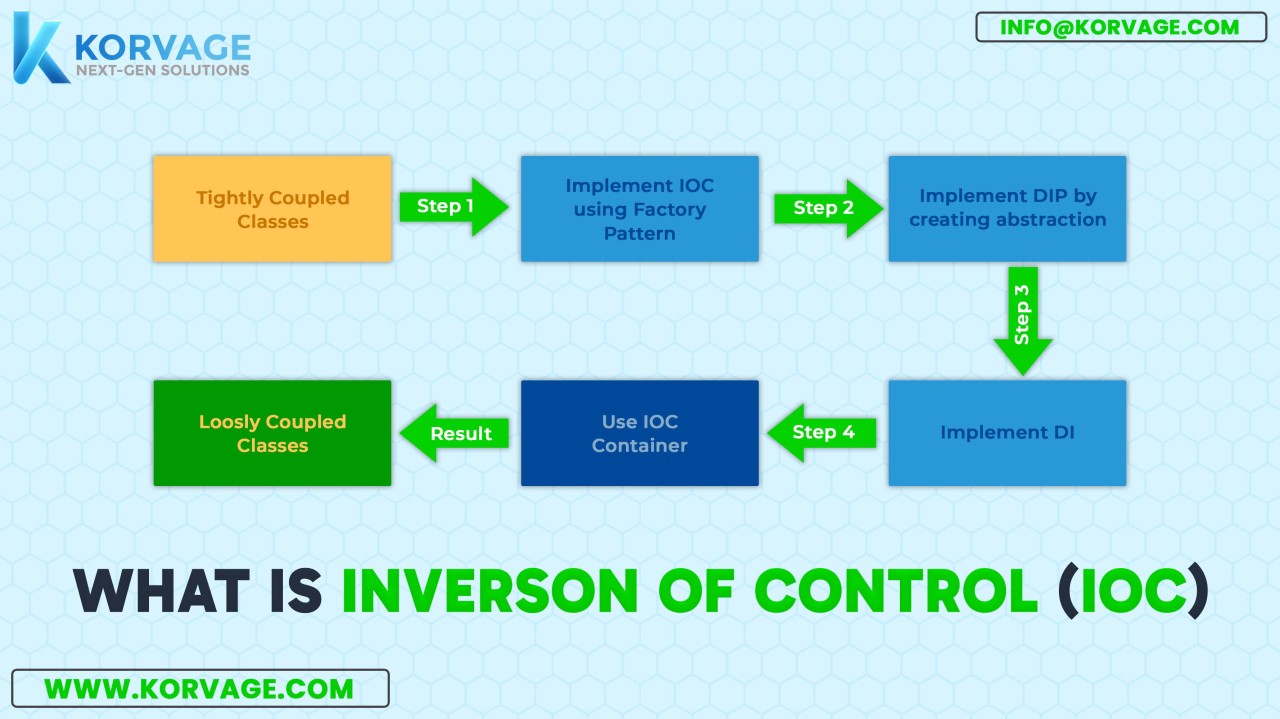
@Autowired

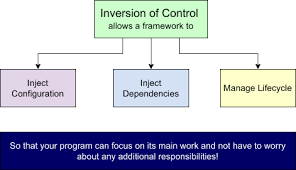
Employee emp;

@SpringBootApplication =

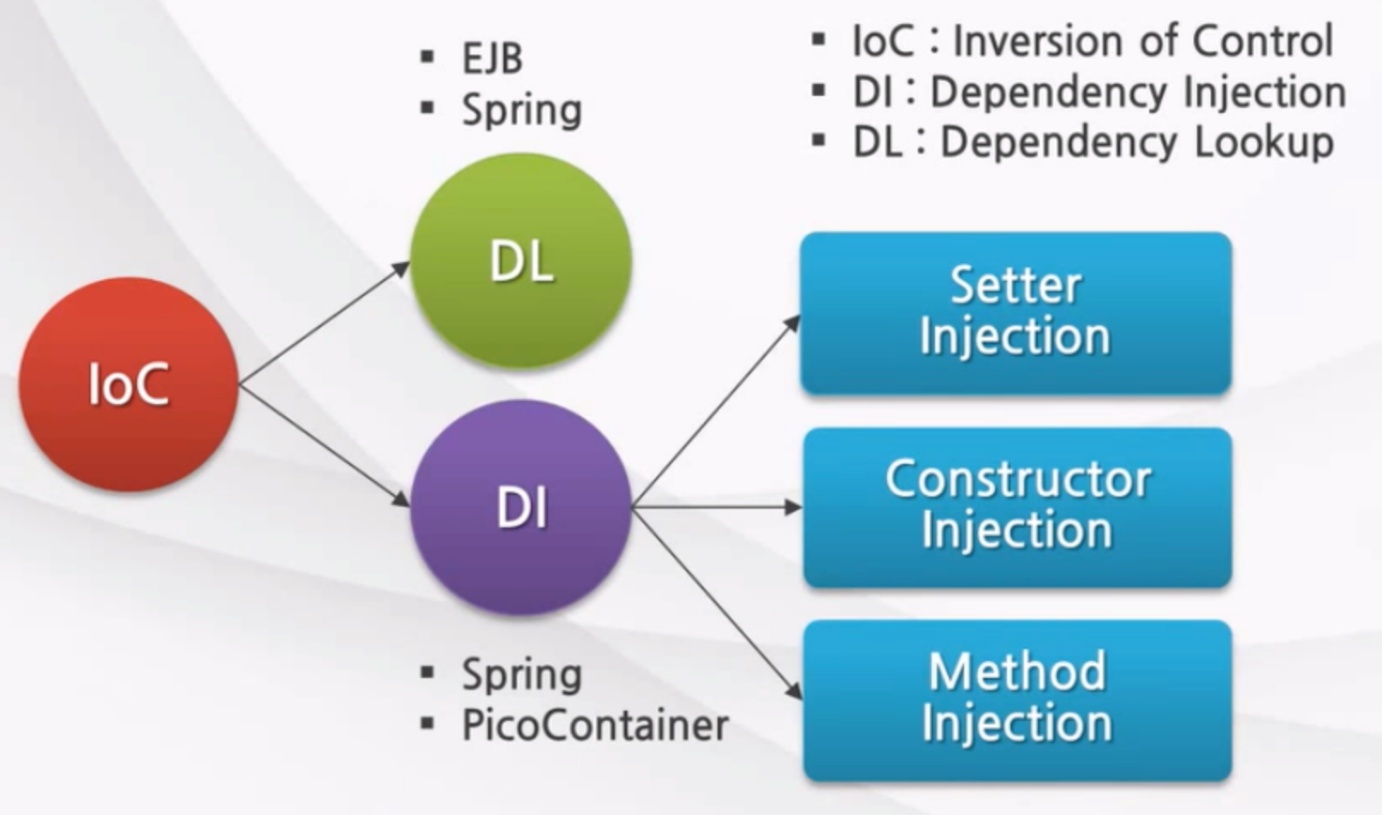
Design Pattern (IoC, DI)

IoC = Inversion of Control.





DI = Dependency Injection.



Spring Components are Managed Spring Beans.

Employee emp = new Employee(); -- Setter Injection

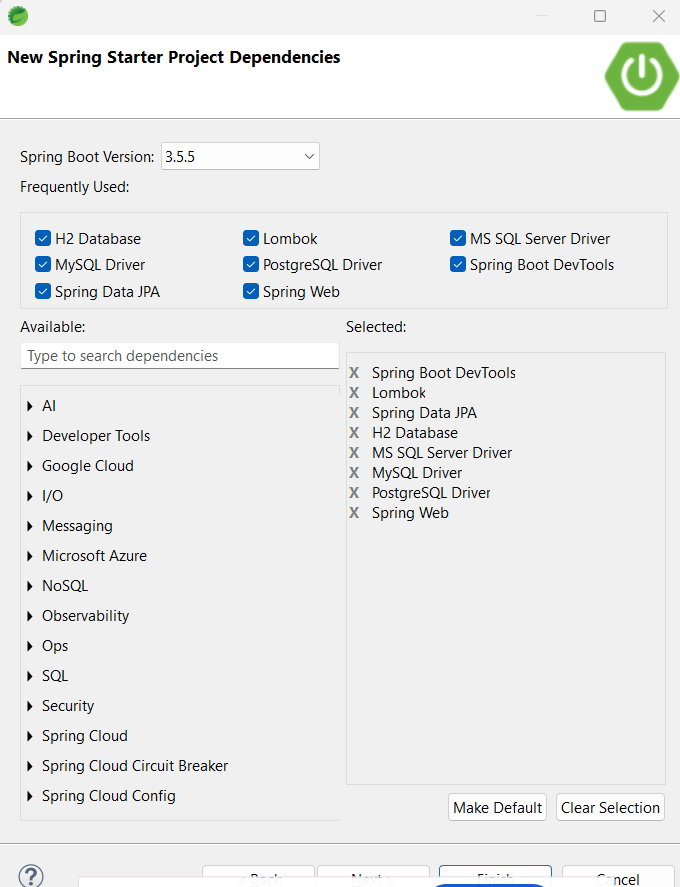
Employee emp1 = new Employee(100,”test”,”[test@gmail.com](mailto:test@gmail.com)”); -- Constructor Injection (recommended approach)

Spring uses the power of POJO. (Plain Old Java Object) [A Simple class where it’s not extending other class nor implementing any interfaces]

With the help of Annotations Spring provides more functionality to POJO Classes.

CLI version of ARS\_V1 (Airline Reservation System)

ARS\_V2 (Spring Boot Monolith Application)



**package** com.example.ars.controller;

**import** java.util.List;

**import** java.util.Optional;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.DeleteMapping;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.PutMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.example.ars.model.User;

**import** com.example.ars.service.UserService;

@RestController

@RequestMapping("/api/v1/users")

**public** **class** UserController {

**private** UserService userService;

**public** UserController(UserService userService) {

**super**();

**this**.userService = userService;

}

@GetMapping

**public** List<User> findAll() {

**return** userService.findAll();

}

@GetMapping("/{id}")

**public** Optional<User> findById(@PathVariable Integer id) {

**return** userService.findById(id);

}

@GetMapping("/email/{email}")

**public** Optional<User> findByEmail(@PathVariable String email) {

**return** userService.findByEmail(email);

}

@GetMapping("/mobile/{mobile}")

**public** Optional<User> findByMobile(@PathVariable Long mobile) {

**return** userService.findByMobile(mobile);

}

@GetMapping("/username/{username}")

**public** Optional<User> findByUsername(@PathVariable String username) {

**return** userService.findByUsername(username);

}

@PostMapping

**public** User save(@RequestBody User user) {

**return** userService.save(user);

}

@PutMapping("/{id}")

**public** User update(@PathVariable Integer id, @RequestBody User user) {

**return** userService.update(id,user);

}

@DeleteMapping("/{id}")

**public** **void** delete(@PathVariable Integer id) {

userService.delete(id);

}

}

**package** com.example.ars.service;

**import** java.util.List;

**import** java.util.Optional;

**import** org.springframework.stereotype.Service;

**import** com.example.ars.model.User;

**import** com.example.ars.repo.UserRepository;

@Service

**public** **class** UserService {

**private** UserRepository userRepository;

**public** UserService(UserRepository userRepository) {

**super**();

**this**.userRepository = userRepository;

}

**public** List<User> findAll() {

**return** userRepository.findAll();

}

**public** Optional<User> findById(Integer id) {

**return** userRepository.findById(id);

}

**public** User save(User user) {

**return** userRepository.save(user);

}

**public** User update(Integer id, User user) {

**return** userRepository.save(user);

}

**public** **void** delete(Integer id) {

userRepository.deleteById(id);

}

**public** Optional<User> findByEmail(String email){

**return** userRepository.findByEmail(email);

}

**public** Optional<User> findByUsername(String username){

**return** userRepository.findByUsername(username);

}

**public** Optional<User> findByMobile(Long mobile){

**return** userRepository.findByMobile(mobile);

}

}

**package** com.example.ars.repo;

**import** java.util.Optional;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.example.ars.model.User;

**public** **interface** UserRepository **extends** JpaRepository<User, Integer> {

Optional<User> findByEmail(String email);

Optional<User> findByMobile(Long mobile);

Optional<User> findByUsername(String username);

}

**package** com.example.ars.model;

**import** java.time.LocalDateTime;

**import** jakarta.persistence.Column;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.GeneratedValue;

**import** jakarta.persistence.GenerationType;

**import** jakarta.persistence.Id;

**import** jakarta.persistence.Table;

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

//Normal Bean Class

//Entity Bean Class

@Entity

@Table(name = "users")

@Data

@NoArgsConstructor

@AllArgsConstructor

**public** **class** User {

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

**private** Integer id;

@Column(nullable = **false**, unique = **true**)

**private** String username;

**private** String hashedPassword;

@Column(unique = **true**)

**private** String email;

@Column(unique = **true**)

**private** Long mobile;

//Audit Logs/ Fields/properties

**private** LocalDateTime createdAt;

**private** Integer createdBy;

**private** LocalDateTime updatedAt;

**private** Integer updatedBy;

}