ConsoleBased application – courseapplication

1.Insert Formula in SpringbootTo create a Console based application we have to implements CommandlineRunner Interface in main class and override run() method in the main method.

2. In the console based application, our application starts from main method but not in web based or RestAPI’s.

3. We are having Service package in that Icourse interface is there which has abstract methods

getlistofcourse()

getcoursebyID()

getcoursebyType()

package com.courseapp.service;  
  
import java.util.List;  
  
import com.courseapp.model.Course;  
  
public interface ICourseService {  
  
 List<Course> getAll();  
 Course getCourseById(int CourseId);  
 List<Course> CourseByType(String CourseType);  
 List<Course> getByLessDurationAndCost(int duration,int cost);  
 List<String> getCourseName();  
}

4. Going to implement the CourseServiceImpl class Icourse Interface

5. Having Util package. This package has CourseUtil class which is going to have list of courses. We don’t have repository package. Instead of db we have util class which has list of courses.

6. @Service and @Component both are stereotype annotations. There are no differences between them. Both create object for the class. Service layer is used to provide business logic. For easy understand and maintenance we have provide here @Service.

7. Here Lombok dependency is used to reduce boilerplatecode. (Getter and Setter methods, Noarg Constructor, AllArg constructor, toString() Method everything is given in single line)

@NoArgsConstructor

@AllArgsConstructor

@Data

Data is a combination of Getter, Setter, toString(),hashcode() and equals()

8. Create Model class with following instance variable. Here CourseID going to work through the internet to DB as a primary key. Primitive data types will not work in the internet so have to give wrapper class Object always for primary Key.

Course.java

**package** com.courseapp.model;

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.Getter;

**import** lombok.NoArgsConstructor;

**import** lombok.Setter;

**import** lombok.ToString;

@NoArgsConstructor

@AllArgsConstructor

@Data

//@Getter

//@Setter

//@ToString

**public** **class** Course {

**private** String courseName;

**private** Integer courseId;

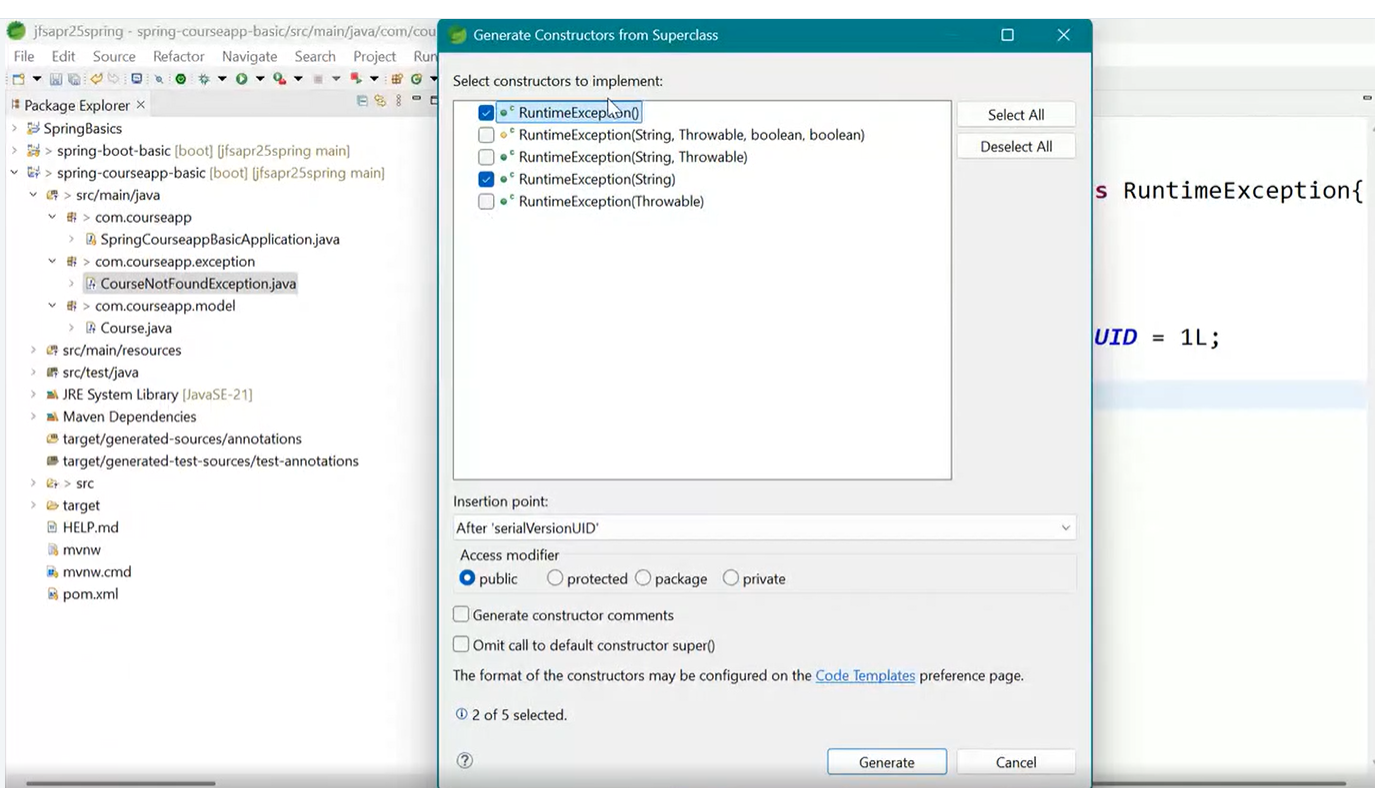
**private** String type; //backend, frontend, cloud,AI

**private** **double** cost;

**private** **int** durationInWeeks;

}

9. Create Exception Package under the package create ClassNotFoundException class file. Always extend RuntimeException because Spring supports unchecked Exception. And keep version ID for reference.



It will create default constructor and parameterized constructor.

CourseNotFoundException.java

**package** com.courseapp.exception;

**public** **class** CourseNotFoundException **extends** RuntimeException{

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** CourseNotFoundException() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** CourseNotFoundException(String message) {

**super**(message);

// **TODO** Auto-generated constructor stub

}

}

10.Enum- allows to add only fixed set of elements.

**package** com.courseapp.model;

**public** **enum** CourseType {

***FRONTEND***,***BACKEND***,***AI***,***CLOUDCOMPUTING***

}

//Enum- allows to add only fixed set of elements.

11.Create CourseUtil class to provide courses because it is consolebased application

**package** com.courseapp.util;

**import** java.util.Arrays;

**import** java.util.List;

**import** org.springframework.stereotype.Component;

**import** com.courseapp.model.Course;

**import** com.courseapp.model.CourseType;

@Component

**public** **class** CourseUtil {

**public** List<Course> getAllCourses() {

**return** Arrays.*asList*(**new** Course("Java", 1, CourseType.***BACKEND***.name(), 2000, 4),

**new** Course("Spring", 2, CourseType.***BACKEND***.name(), 5000, 5),

**new** Course("React", 3, CourseType.***FRONTEND***.name(), 6000, 4),

**new** Course("Python", 4, CourseType.***AI***.name(), 5000, 3),

**new** Course("AWS", 5, CourseType.***CLOUDCOMPUTING***.name(), 12000, 5),

**new** Course("GCP", 6, CourseType.***CLOUDCOMPUTING***.name(), 14000, 10),

**new** Course("AZURE", 7, CourseType.***CLOUDCOMPUTING***.name(), 13000, 10),

**new** Course("Microservices", 8, CourseType.***BACKEND***.name(), 10000, 6));

}

}

12. CourseServiceImpl class depends on CourseUtil. So we have to create reference and inject the dependency by using setter based DI and @AutoWired.

package com.courseapp.service;  
  
import java.util.List;  
import java.util.Optional;  
import java.util.stream.Collectors;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import com.courseapp.exception.CourseNotFoundException;  
import com.courseapp.model.Course;  
import com.courseapp.util.CourseUtil;  
  
@Service  
public class CourseServiceImpl implements ICourseService {  
  
 private CourseUtil courseutil;  
  
 @Autowired  
 public void setCourseutil(CourseUtil courseutil) {  
 this.courseutil = courseutil;  
 }  
  
 @Override  
 public List<Course> getAll() {  
 return courseutil.getAllCourses();  
 }  
  
 @Override  
 public Course getCourseById(int courseId) {  
 List<Course> courses = courseutil.getAllCourses();  
// List<Course> courses = courseUtil.getAllCourses();  
// Optional<Course> courseopt = courses.stream()  
// .filter(course->course.getCourseId()==courseId)  
// .findFirst();  
// if(courseopt.isPresent())  
// return courseopt.get();  
// return null;  
 return courses.stream().filter(course -> course.getCourseId() == courseId).findFirst()  
 .orElseThrow(() -> new CourseNotFoundException("invalid id"));  
 }  
  
 @Override  
 public List<Course> CourseByType(String type) {  
 List<Course> courses = courseutil.getAllCourses();  
 List<Course> ncourses = courses.stream().filter(course -> course.getType().equals(type)).toList();  
 if (ncourses.isEmpty())  
 throw new CourseNotFoundException("type not found");  
 return ncourses;  
  
 }  
  
 @Override  
 public List<Course> getByLessDurationAndCost(int duration, int cost) {  
 List<Course> courses = courseutil.getAllCourses();  
  
 List<Course> ncourses = courses.stream()  
 .filter(course -> (course.getCost() <= cost) && (course.getDurationInWeeks() <= duration)).toList();  
 if (ncourses.isEmpty())  
 throw new CourseNotFoundException("There is no course available in less duration and less cost");  
 return ncourses;  
 }  
  
 @Override  
 public List<String> getCourseName() {  
 List<Course> courses = courseutil.getAllCourses();  
 List<String> ncourses = courses.stream().map((course) -> course.getCourseName()).collect(Collectors.*toList*());  
  
 if (ncourses.isEmpty())  
 throw new CourseNotFoundException("No course available");  
 return ncourses;  
 }  
  
}

13. main class:

Implemented CommandLineRunner interface and run()

Autowired IcourseService.

package com.courseapp;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
import com.courseapp.service.ICourseService;  
  
@SpringBootApplication  
public class SpringCourseappBasicApplication implements CommandLineRunner {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SpringCourseappBasicApplication.class, args);  
 }  
 @Autowired  
 private ICourseService iCourseService;  
 @Override  
 public void run(String... args) throws Exception {  
 System.*out*.println("\*\*\*\*\*\*\*\*Listing all the courses\*\*\*\*\*\*\*\*");  
 iCourseService.getAll().forEach(System.*out*::println);  
 System.*out*.println("\*\*\*\*\*\*\*\*Listing the course by ID\*\*\*\*\*\*\*\*");  
 System.*out*.println(iCourseService.getCourseById(1));  
 System.*out*.println("\*\*\*\*\*\*\*\*Listing the course by Type\*\*\*\*\*\*\*\*");  
 iCourseService.CourseByType("BACKEND").forEach(System.*out*::println);  
 System.*out*.println("\*\*\*\*\*\*\*\*Listing the course by less duration and cost\*\*\*\*\*\*\*\*");  
 System.*out*.println(iCourseService.getByLessDurationAndCost(10, 10000));  
 System.*out*.println("\*\*\*\*\*\*\*\*Listing the coursenames alone\*\*\*\*\*\*\*\*");  
 System.*out*.println(iCourseService.getCourseName());  
 }  
  
}

14. Output:

. \_\_\_\_ \_ \_\_ \_ \_

/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )

' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

=========|\_|==============|\_\_\_/=/\_/\_/\_/

:: Spring Boot :: (v3.5.3)

2025-06-28T00:26:31.174-04:00 INFO 13516 --- [spring-courseapp-basic] [ main] c.c.SpringCourseappBasicApplication : Starting SpringCourseappBasicApplication using Java 23.0.1 with PID 13516 (I:\Spring\SpringBoot\SpringBoot\_Learning\spring-courseapp-basic\target\classes started by ramya in I:\Spring\SpringBoot\SpringBoot\_Learning\spring-courseapp-basic)

2025-06-28T00:26:31.176-04:00 INFO 13516 --- [spring-courseapp-basic] [ main] c.c.SpringCourseappBasicApplication : No active profile set, falling back to 1 default profile: "default"

2025-06-28T00:26:31.712-04:00 INFO 13516 --- [spring-courseapp-basic] [ main] c.c.SpringCourseappBasicApplication : Started SpringCourseappBasicApplication in 0.913 seconds (process running for 1.298)

\*\*\*\*\*\*\*\*Listing all the courses\*\*\*\*\*\*\*\*

Course(courseName=Java, courseId=1, type=BACKEND, cost=2000.0, durationInWeeks=4)

Course(courseName=Spring, courseId=2, type=BACKEND, cost=5000.0, durationInWeeks=5)

Course(courseName=React, courseId=3, type=FRONTEND, cost=6000.0, durationInWeeks=4)

Course(courseName=Python, courseId=4, type=AI, cost=5000.0, durationInWeeks=3)

Course(courseName=AWS, courseId=5, type=CLOUDCOMPUTING, cost=12000.0, durationInWeeks=5)

Course(courseName=GCP, courseId=6, type=CLOUDCOMPUTING, cost=14000.0, durationInWeeks=10)

Course(courseName=AZURE, courseId=7, type=CLOUDCOMPUTING, cost=13000.0, durationInWeeks=10)

Course(courseName=Microservices, courseId=8, type=BACKEND, cost=10000.0, durationInWeeks=6)

\*\*\*\*\*\*\*\*Listing the course by ID\*\*\*\*\*\*\*\*

Course(courseName=Java, courseId=1, type=BACKEND, cost=2000.0, durationInWeeks=4)

\*\*\*\*\*\*\*\*Listing the course by Type\*\*\*\*\*\*\*\*

Course(courseName=Java, courseId=1, type=BACKEND, cost=2000.0, durationInWeeks=4)

Course(courseName=Spring, courseId=2, type=BACKEND, cost=5000.0, durationInWeeks=5)

Course(courseName=Microservices, courseId=8, type=BACKEND, cost=10000.0, durationInWeeks=6)

\*\*\*\*\*\*\*\*Listing the course by less duration and cost\*\*\*\*\*\*\*\*

[Course(courseName=Java, courseId=1, type=BACKEND, cost=2000.0, durationInWeeks=4), Course(courseName=Spring, courseId=2, type=BACKEND, cost=5000.0, durationInWeeks=5), Course(courseName=React, courseId=3, type=FRONTEND, cost=6000.0, durationInWeeks=4), Course(courseName=Python, courseId=4, type=AI, cost=5000.0, durationInWeeks=3), Course(courseName=Microservices, courseId=8, type=BACKEND, cost=10000.0, durationInWeeks=6)]

\*\*\*\*\*\*\*\*Listing the coursenames alone\*\*\*\*\*\*\*\*

[Java, Spring, React, Python, AWS, GCP, AZURE, Microservices]

Process finished with exit code 0