

Quiz Questions

(1) What is DI?

← Classroom

Dependency Injection in Spring boot
What is Dependency Injection

?

Problem Submissions Doubts

What is DI
Easy • Score 20/20

Send feedback

Problem statement
What is Dependency Injection?

Options: Pick one correct answer from below

☐ A method for dynamically generating code in an application

☒ A pattern for resolving dependencies between objects

☐ A way to create objects with fewer dependencies

☐ A technique for improving the performance of an application

Solution description

Dependency Injection (DI) is a design pattern that resolves dependencies between objects. It's a way to separate an object's construction and use from its dependencies so that the object is not tightly coupled to its dependencies. DI often creates more flexible, testable, and maintainable code.

(2) IOC Implementation

← Classroom

Dependency Injection in Spring boot
IOC

?

Problem Submissions Doubts

IOC implementation I
Easy • Score 20/20

Send feedback

Problem statement
You are given an Employee interface, with Manager and Engineer as its implementations as follows:

```
// Employee Interface
interface Employee {
    void display();
}

// Manager Class
class Manager implements Employee {
    void display() {
        System.out.println("This is a Manager");
    }
}

// Engineer Class
class Engineer implements Employee {
    void display() {
        System.out.println("This is an Engineer");
    }
}
```

Options: Pick one correct answer from below

☐ A)

☐ B)

☐ C)

☒ D)

Solution description

- IOC allows us to decouple objects and delegate their creation and management to the framework. The correct steps to implement IOC are A, B, and C. In step A, we configure the Spring beans in the XML file. In step B, we create a Spring container to retrieve the beans. In step C, we retrieve the bean from the container.
- Step D creates a new instance of the Manager class using the new operator, which is not in line with the concept of IOC. Therefore, step D is NOT required for implementing inversion of control.

Given below are the steps for implementing IOC with an XML file. Which action is NOT required for implementing inversion of control from the following?

A) Configure spring beans in the application.xml

```
<bean id="manager" class="com.example.Manager"/>
<bean id="engineer" class="com.example.Engineer"/>
```

B) Create a Spring Container

```
ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("application.xml");
```

C) Retrieve Bean from Container

```
Employee manager = (Employee) context.getBean("manager");
```

D) Create object from Class

```
Manager manager = new Manager();
```

(3) IOC Error Analysis

← Classroom

Dependency Injection in Spring boot
IOC II

?

V

Problem Submissions Doubts

✓ IOC - Error Analysis - Dependency Injection
Easy • Score 20/20

Problem statement [Send feedback](#)
Continuing the previous question, here is the main class.

```

// Following is the MainApp.java file -

package com.EmployeeDemo
import org.springframework.context.ApplicationContext;

public class MainApp {
    public static void main(String[] args) {
        ClassPathXmlApplicationContext context = new
        ClassPathXmlApplicationContext("applicationContext.xml");

        Employee engineer = context.getBean("employee");
        engineer.display();
    }
}

```

The above code shows the following error on the compilation

```

java: incompatible types: java.lang.Object cannot be
converted to com.EmployeeDemo.Employee

```

What is the problem with the code?

Options: Pick one correct answer from below

☐ The ClassPathXmlApplicationContext class is not a Spring Boot-compatible class.

☐ The applicationContext.xml file is not located in the classpath.

☒ The engineer bean is not typecasted to Employee. ✓

☐ The display() method cannot be accessed through the engineer object.

Solution description
When we want to access beans, we use context.getBean() method with the bean id as a string in the method parameter. Also, we need to typecast it to the required type. I.e. Employee, as the getBean() method returns a plain object.

(4) Constructor Injection

← Classroom

Dependency Injection in Spring boot
[What is Constructor Injection](#)

?

V

Problem Submissions Doubts

✓ Constructor_Injection
Easy • Score 20/20

Problem statement [Send feedback](#)
What is constructor injection?

Options: Pick one correct answer from below

☐ A technique for passing dependencies to a class using public fields.

☐ A technique for passing dependencies to a class using setter methods.

☒ A technique for passing dependencies to a class using its constructor. ✓

☐ A technique for passing dependencies to a class using reflection.

Solution description
Constructor injection is a technique for passing dependencies to a class using its constructor. For example -

```

public class Car{
    private Engine engine;

    public Car(Engine engine) {
        This.engine = engine;
    }
    // ...
}

```

(5) Constructor Injection

← Classroom

Dependency Injection in Spring boot
Constructor Injection Advantages

?

V

Problem Submissions Doubts

✓ Constructor Injection

Easy • Score 20/20

Send feedback

Problem statement

What are the advantages of using constructor injection?

Options: Pick one correct answer from below

☐ Constructor injection can help to reduce coupling between classes.

☐ Constructor injection can help to make testing and mocking easier.

☐ Constructor injection can make it easier to manage the lifecycle of objects.

☒ All of the above.

Solution description

- Constructor injection can reduce coupling between classes, and make testing and mocking easier. By passing dependencies through a class's constructor, the class is less tightly coupled to its dependencies, which can make it easier to change or replace those dependencies later without affecting the class itself.
- This also makes it easier to test the class, as you can create mock or fake dependencies to use during testing. Finally, constructor injection can make it easier to manage the lifecycle of objects (we will learn more about the bean life cycle in the next lecture), as you can control when and how dependencies are created and passed to a class.

(6) Error Analysis Character Builder

← Classroom

Dependency Injection in Spring boot
Constructor Injection CharacterBuilder

?

V

Problem Submissions Doubts

✓ Error Analysis CharacterBuilder

Easy • Score 20/20

Send feedback

Problem statement

Scenario: Suppose you're building a video game with a character creation screen. You have a Character class that represents a player's character. The Character class depends on a CharacterBuilder object, which initialises the character's attributes. You want to use Constructor Injection to pass the CharacterBuilder dependency to the Character class.

Which of the following code snippets correctly implements Constructor Injection for the Character class?

Options: Pick one correct answer from below

☐

```
public class Character {
    private final CharacterBuilder characterBuilder;

    public Character() {
        characterBuilder = new CharacterBuilder();
    }
    // ...
}
```

☒

```
public class Character {
    private final CharacterBuilder characterBuilder;

    public Character(CharacterBuilder characterBuilder) {
        this.characterBuilder = characterBuilder;
    }
    // ...
}
```

(7) Setter vs Constructor Injection

← Classroom

Dependency Injection in Spring boot
Constructor vs Setter injection

?

Problem Submissions Doubts

✓ setter vs constructor injection

Easy • Score 20/20

Problem statement

Which of the following is an advantage of Constructor Injection over Setter Injection?

[Send feedback](#)

Options: Pick one correct answer from below

☐ It allows for optional dependencies.

☐ It's faster than Setter Injection

☒ It ensures that an object is fully initialized before used

☐ It is easier to implement.

Solution description

One advantage of Constructor Injection over Setter Injection is that it ensures an object is fully initialised before use. This makes the code more robust and less prone to runtime errors caused by missing dependencies. On the other hand, Setter Injection allows for optional dependencies, which can lead to objects being used before they're fully initialised.

(8) Setter Injection Example

← Classroom

Dependency Injection in Spring boot
Setter Injection Example

?

Problem Submissions Doubts

✓ Setter Injection Example

Easy • Score 20/20

Problem statement

Which of the following is an example of a setter injection?

[Send feedback](#)

☐

```
public class CarPoolingService{
    private Location location;
    private Date date;

    public void setLocation(String location) {
        this.location = new Location(location);
    }
    public void setDate(String date) {
        this.date = new Date(date);
    }
}
```

☒

```
public class CarPoolingService{
    private Location location;
    private Date date;

    public void setLocation(Location location) {
        this.location = location;
    }
    public void setDate(Date date) {
        this.date = date;
    }
}
// ...
```

(9) DI Benefits

←

Classroom

Dependency Injection in Spring boot

DI Benefits

?

Problem

Submissions

Doubts

✓

DI Benefits

Easy • Score 20/20

Problem statement

What are the benefits of using Dependency Injection?

[Send feedback](#)

Options: Pick one correct answer from below

☐ Improved testability and maintainability

☐ Better performance and scalability

☐ Reduced complexity and coupling

☒ All of the above

Solution description

The benefits of using Dependency Injection include improved testability and maintainability, better performance and scalability, and reduced complexity and coupling. By reducing coupling between objects, DI makes code easier to change and less prone to errors.