DURGASOFT

Day-11: What Is @Configuration in Spring?

A comprehensive guide to understanding Spring's powerful @Configuration annotation and its role in modern application development

Understanding @Configuration Annotation

The @Configuration annotation tells Spring Framework that a particular class contains Bean definitions. Instead of relying on traditional XML configuration files, you can write clean, type-safe Java methods using @Bean to create and manage objects throughout your application lifecycle.

This modern approach makes configuration more readable, maintainable, and easier to debug — especially beneficial in Spring Boot cloud projects where configuration complexity can grow rapidly. Type safety ensures compile-time checking, reducing runtime errors significantly.

Key Benefits:

- Type-safe configuration
- Better IDE support
- Easier refactoring
- No XML overhead

@Configuration in Action

Step 1: Annotate Class

Mark your configuration class with @Configuration to tell Spring it contains Bean definitions

Step 2: Define Beans

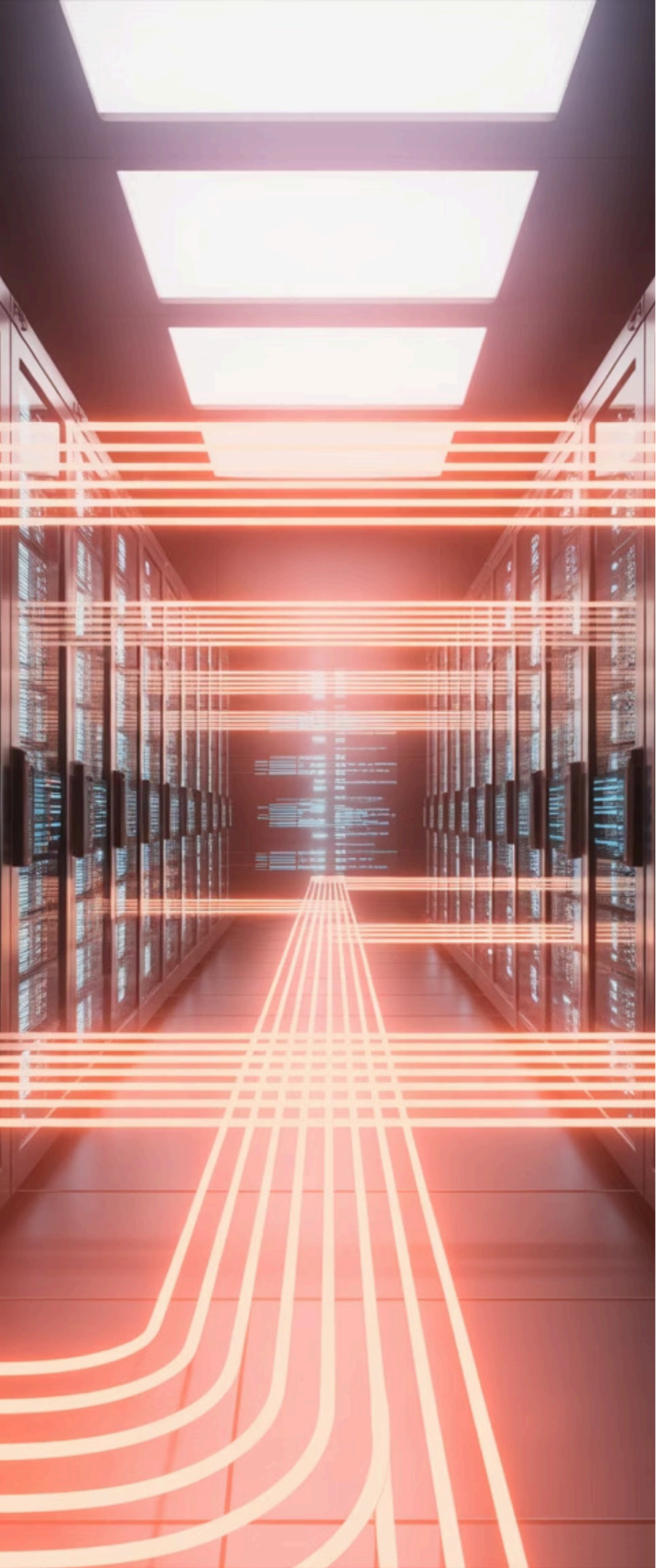
Create methods annotated with @Bean that return the objects you want Spring to manage

Step 3: Spring Creates Beans

Spring automatically reads the class, executes methods, and registers Beans in the container

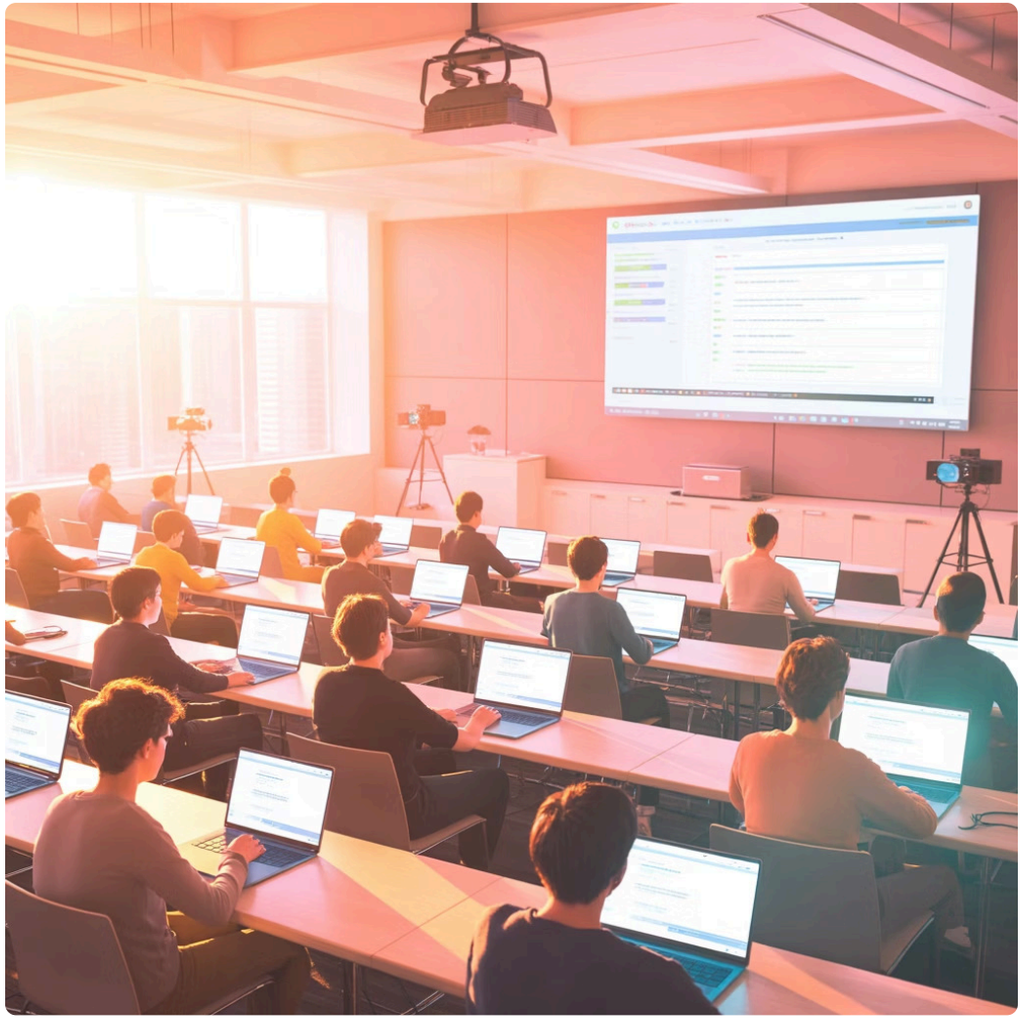
```
@Configuration
public class AppConfig {
    @Bean
    public EmailService emailService() {
        return new EmailService();
    }
}
```

Spring will read this class, run the emailService() method, and create a Bean automatically. This is the modern replacement for old XML configurations, offering superior maintainability and clarity.



When your configuration is clean, your entire application becomes smoother and easier to maintain.

Join DURGASOFT for Spring Framework Training



✨ *If you want online training in SPRING Framework with SPRING BOOT Cloud, join DURGASOFT – India's trusted training institute.*

Contact Numbers

📞 9246212143, 8885252627

Website

🌐 **www.durgasoftonline.com**

Email Us

📩 **durgasoftonlinetraining@gmail.com**

DURGASOFT

Day-12: What Is @Bean in Spring?

Discover how @Bean annotation gives you complete control over object creation and lifecycle management in Spring applications

The Power of @Bean Annotation

The @Bean annotation is used inside a @Configuration class to instruct Spring: "Create this object and manage it as a Bean." This powerful annotation gives you full control over object creation, initialization, and configuration — especially valuable when you need custom setup logic or want to return objects that aren't directly part of your codebase.

Custom Initialization

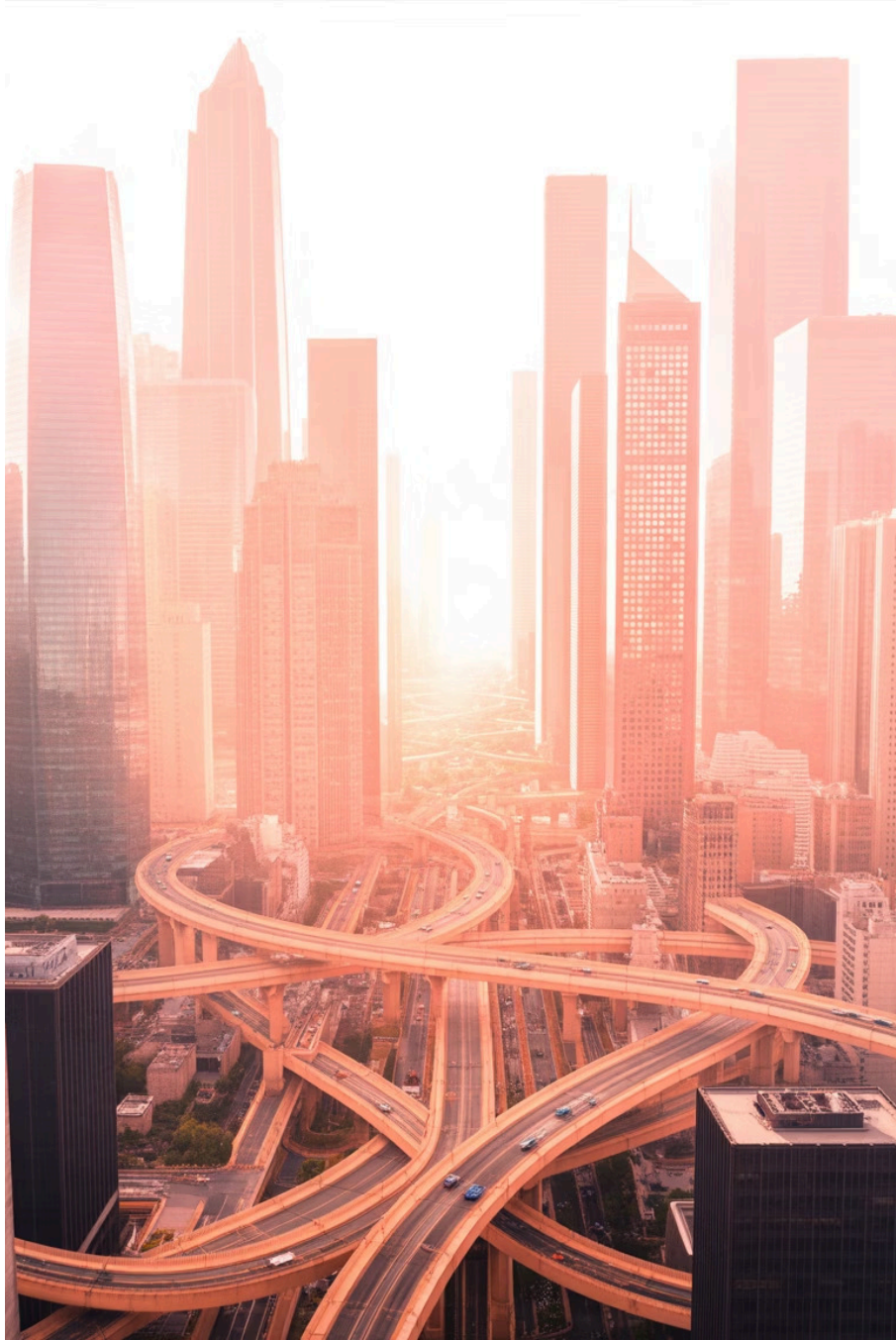
Define complex object creation logic with custom parameters and settings

Third-Party Integration

Manage external library objects that cannot be annotated directly

Configuration Control

Fine-tune Bean properties and dependencies programmatically



@Bean Configuration Example

```
@Configuration
public class AppConfig {

    @Bean
    public DataSource dataSource() {
        return new HikariDataSource();
    }
}
```

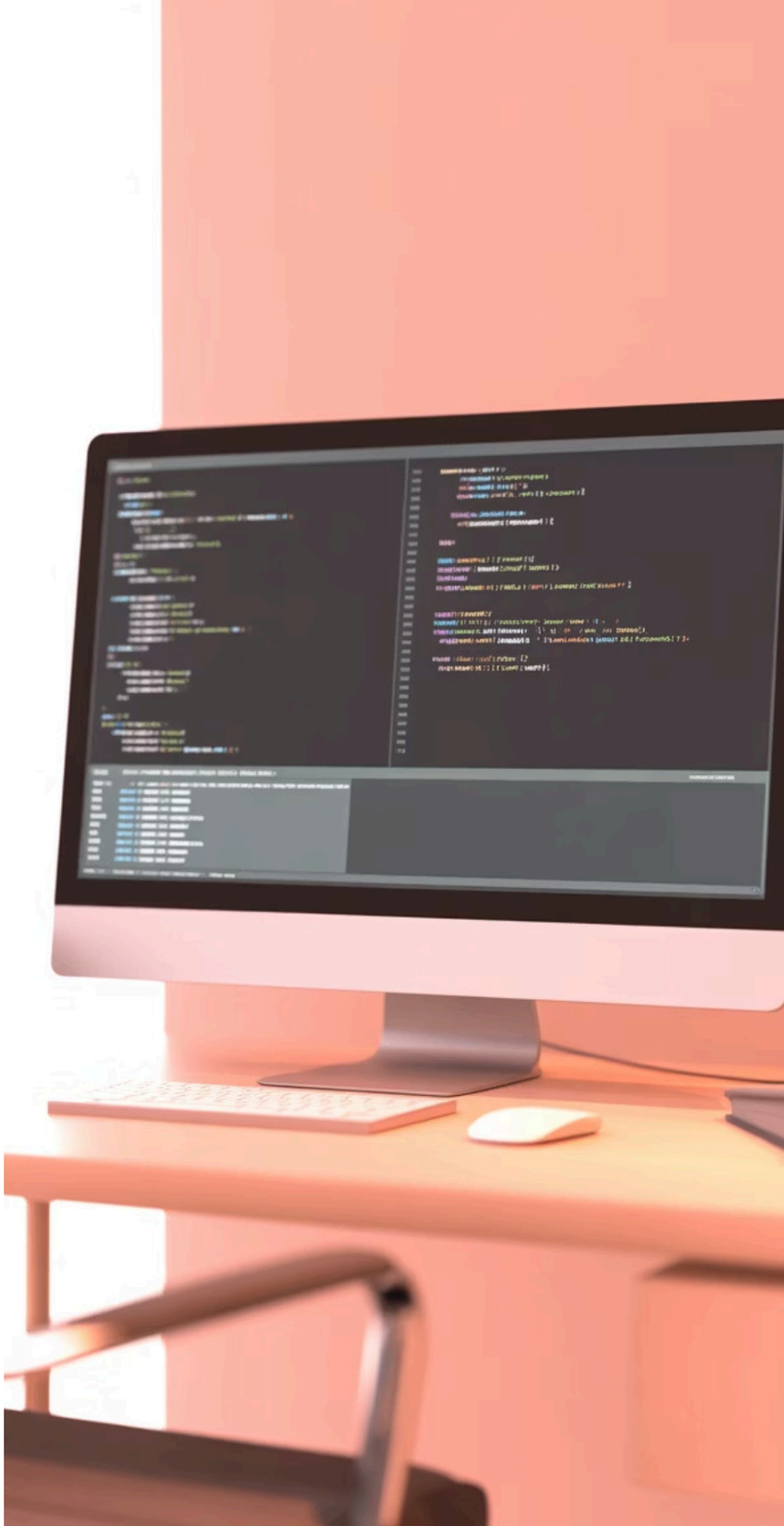
In this example, Spring automatically creates and registers the DataSource Bean in the application context. The method name becomes the Bean identifier by default, though you can customize this.

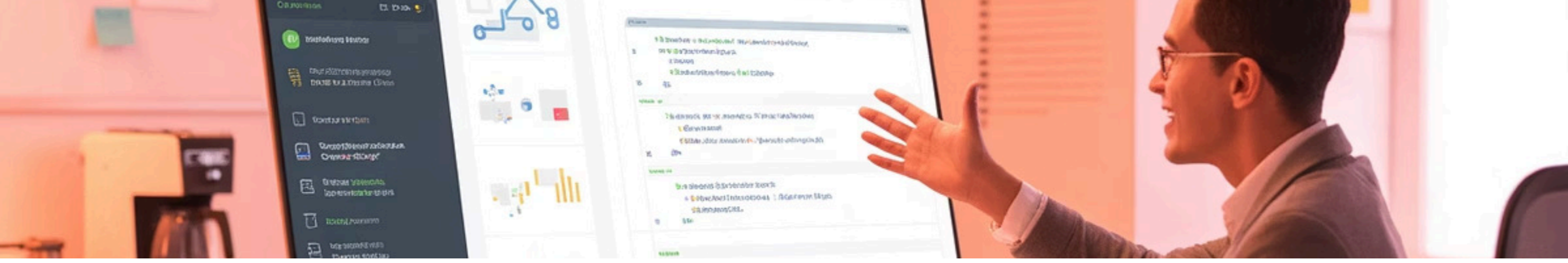
Common Use Cases

- DataSource configuration
- RestTemplate setup
- ObjectMapper customization
- Security components
- Cache managers
- Message converters

This approach is extremely useful for configuring essential services like DataSource, RestTemplate, ObjectMapper, and other infrastructure components that require specific initialization parameters or custom configuration logic.

Great developers don't write more code, they write cleaner and smarter code.





DURGASOFT

Transform Your Spring Skills with DURGASOFT

Call Us Today

📞 9246212143
📞 8885252627

Visit Our Website

🌐 www.durgasoftonline.com

Email Inquiry

📩 durgasoftonlinetraining@gmail.com

✨ *Join DURGASOFT for comprehensive online training in SPRING Framework with SPRING BOOT Cloud – India's most trusted training institute for Java professionals.*

DURGASOFT

Day-13: What Is @ComponentScan in Spring?

Learn how @ComponentScan helps Spring automatically discover and register components in your application

Understanding @ComponentScan

The @ComponentScan annotation tells Spring Framework exactly where to search for classes annotated with @Component, @Service, @Controller, and @Repository. Without this critical annotation, Spring will not detect and create Beans automatically, leaving your application unable to wire dependencies properly.

This annotation is fundamental for component discovery in large projects. It enables Spring to find your code, analyze annotations, and automatically create Bean instances, saving you from manual configuration and reducing boilerplate code significantly.

01

Define Base Packages

Specify which packages Spring should scan for components

02

Spring Scans Classes

Framework examines all classes in specified packages

03

Detect Annotations

Identifies @Component, @Service, @Repository, @Controller

04

Create Beans

Automatically instantiates and registers detected components

@ComponentScan Configuration

```
@Configuration
@ComponentScan(basePackages = "com.durgasoft.app")
public class AppConfig {
    // Configuration logic here
}
```

With this configuration, Spring now scans the com.durgasoft.app package and all its sub-packages, automatically creating Beans for any classes marked with stereotype annotations. This is absolutely essential for any Spring Boot or Spring Core application to work smoothly and efficiently.

@Component

Generic stereotype for any Spring-managed component

@Service

Business logic layer components

@Repository

Data access layer components

@Controller

Web layer MVC controllers

When Spring knows where to look, your entire application falls perfectly into place.

Master Spring Framework with Expert Training

✨ *If you want online training in SPRING Framework with SPRING BOOT Cloud, join DURGASOFT – India's trusted training institute with proven track record.*

Get Started Today

📞 **Contact:** 9246212143, 8885252627

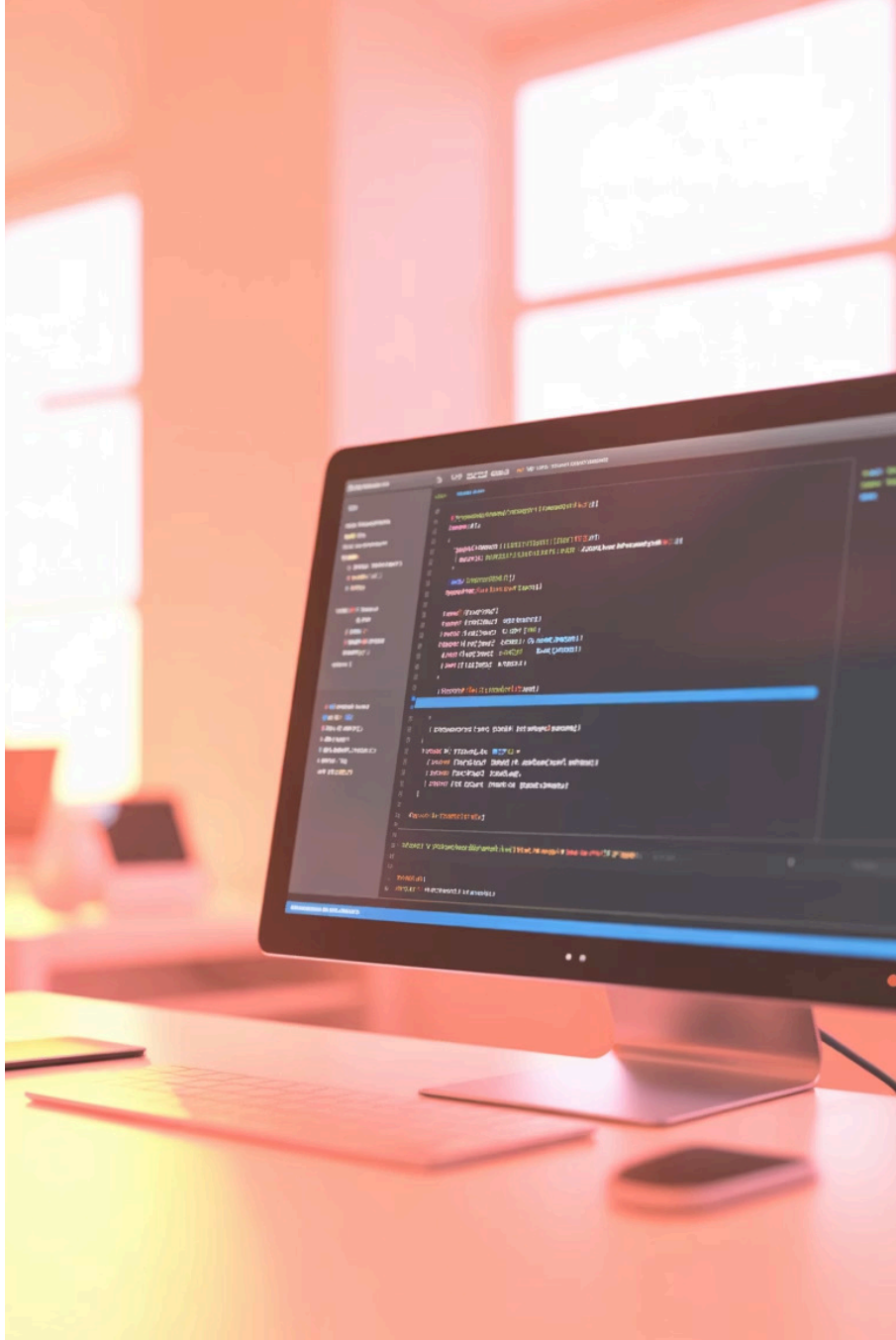
🌐 **Website:** www.durgasoftonline.com

📩 **Email:** durgasoftonlinetraining@gmail.com

DURGASOFT

Day-14: What Is @Value Annotation in Spring?

Explore how @Value annotation enables flexible configuration management and externalized properties in Spring applications



DURGASOFT

@Value Annotation Explained

The @Value annotation is used to inject values from properties files, YAML files, environment variables, or even hard-coded strings directly into your Spring Beans. This powerful feature helps you separate configuration data from business logic, making your application more flexible, maintainable, clean, and remarkably easy to deploy across different cloud environments.

Properties Files

Load values from application.properties or application.yml files

Environment Variables

Inject system environment variables for cloud deployments

Default Values

Specify fallback values using colon syntax

@Value in Practice

Java Code

```
@Value("${app.message}")
private String message;

@Value("${app.timeout:5000}")
private int timeout;

@Value("${app.enabled:true}")
private boolean enabled;
```

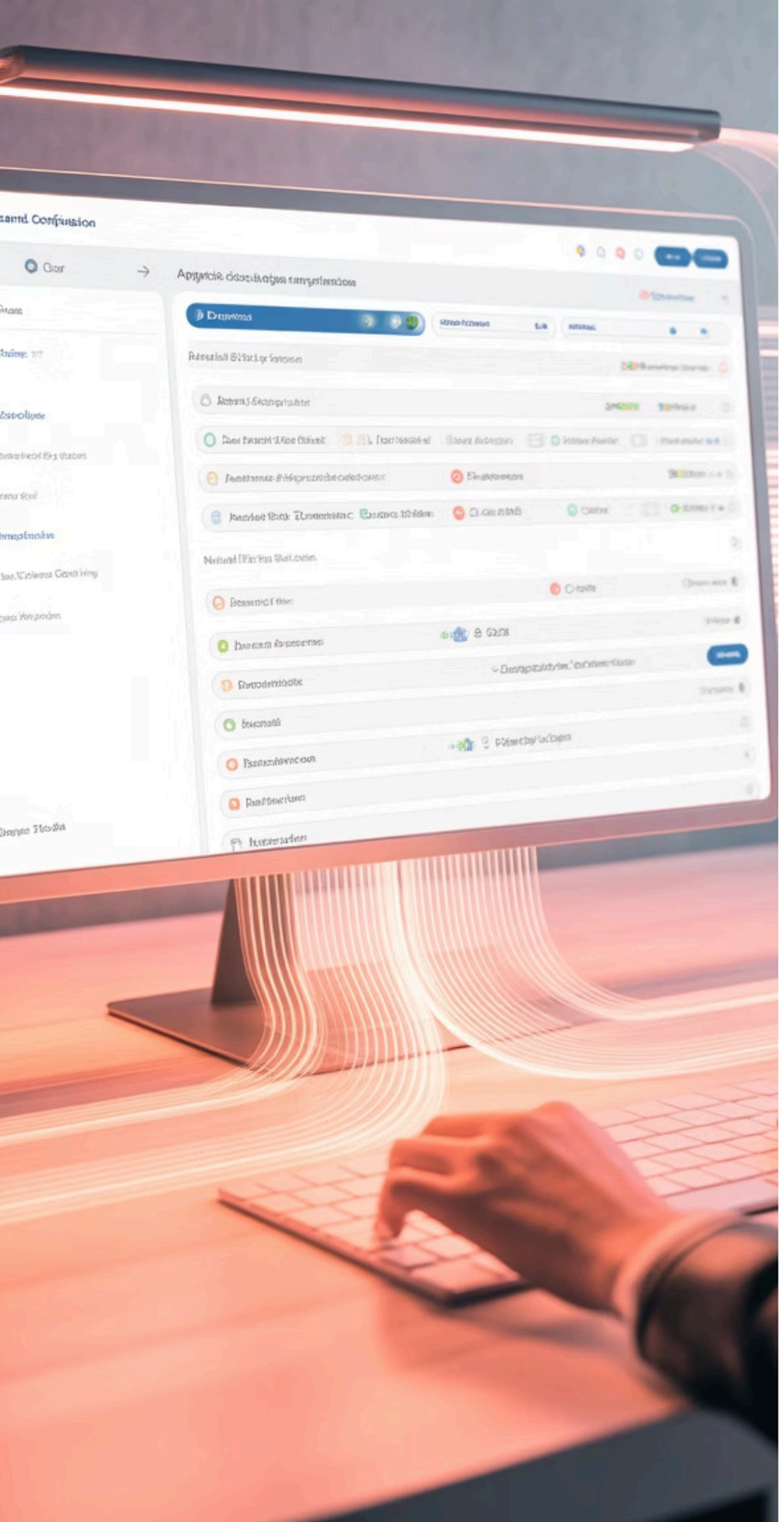
Key Features:

- SpEL (Spring Expression Language) support
- Default value syntax with colon
- Type conversion automatic
- Environment-specific configs

Properties File

```
app.message=Welcome to DURGASOFT
app.timeout=3000
app.enabled=true
```

Spring automatically injects these values at runtime. This approach is extremely useful in cloud applications where configuration values change per environment (development, testing, production), enabling seamless deployment without code changes.



Clean configuration makes your application stronger, faster, and easier to manage.

Advance Your Career with DURGASOFT

Comprehensive Curriculum

Complete Spring Framework and Spring Boot Cloud training designed for real-world projects

Expert Instructors

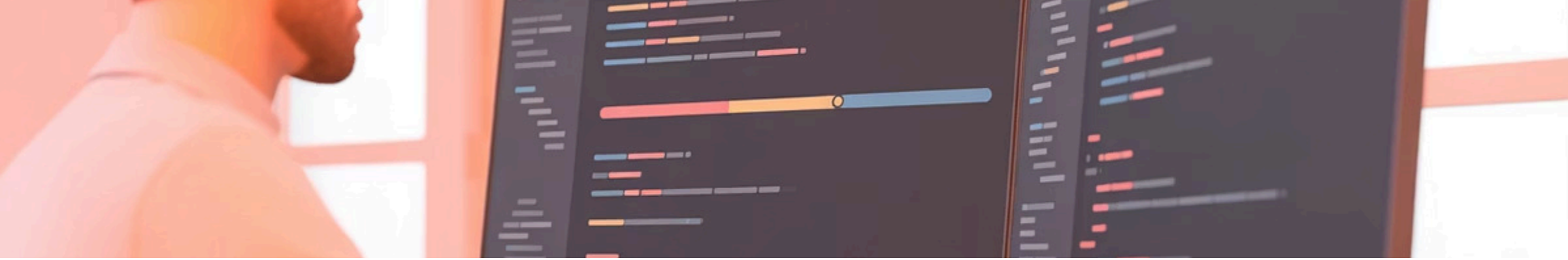
Learn from industry professionals with years of practical experience

Flexible Learning

Online training that fits your schedule with recorded sessions

✨ *Join DURGASOFT – India's trusted training institute for Spring Framework excellence.*

📞 **Contact:** 9246212143, 8885252627 | 🌐 www.durgasoftonline.com | 📩 durgasoftonlinetraining@gmail.com



DURGASOFT

Day-15: What Is @Lazy Annotation in Spring?

Understand how @Lazy annotation optimizes application startup and resource utilization

DURGASOFT

Lazy Initialization Explained

The @Lazy annotation instructs Spring to create a Bean **only when it is actually needed**, instead of eagerly creating it at application startup. This smart approach helps improve startup performance significantly and saves valuable memory resources. It proves especially useful in large Spring Boot and cloud applications where many Beans exist but only a few are used immediately after startup.

Faster Startup

Application launches quicker by deferring Bean creation

Memory Savings

Unused Beans don't consume memory until required

Resource Efficiency

System resources allocated only when necessary

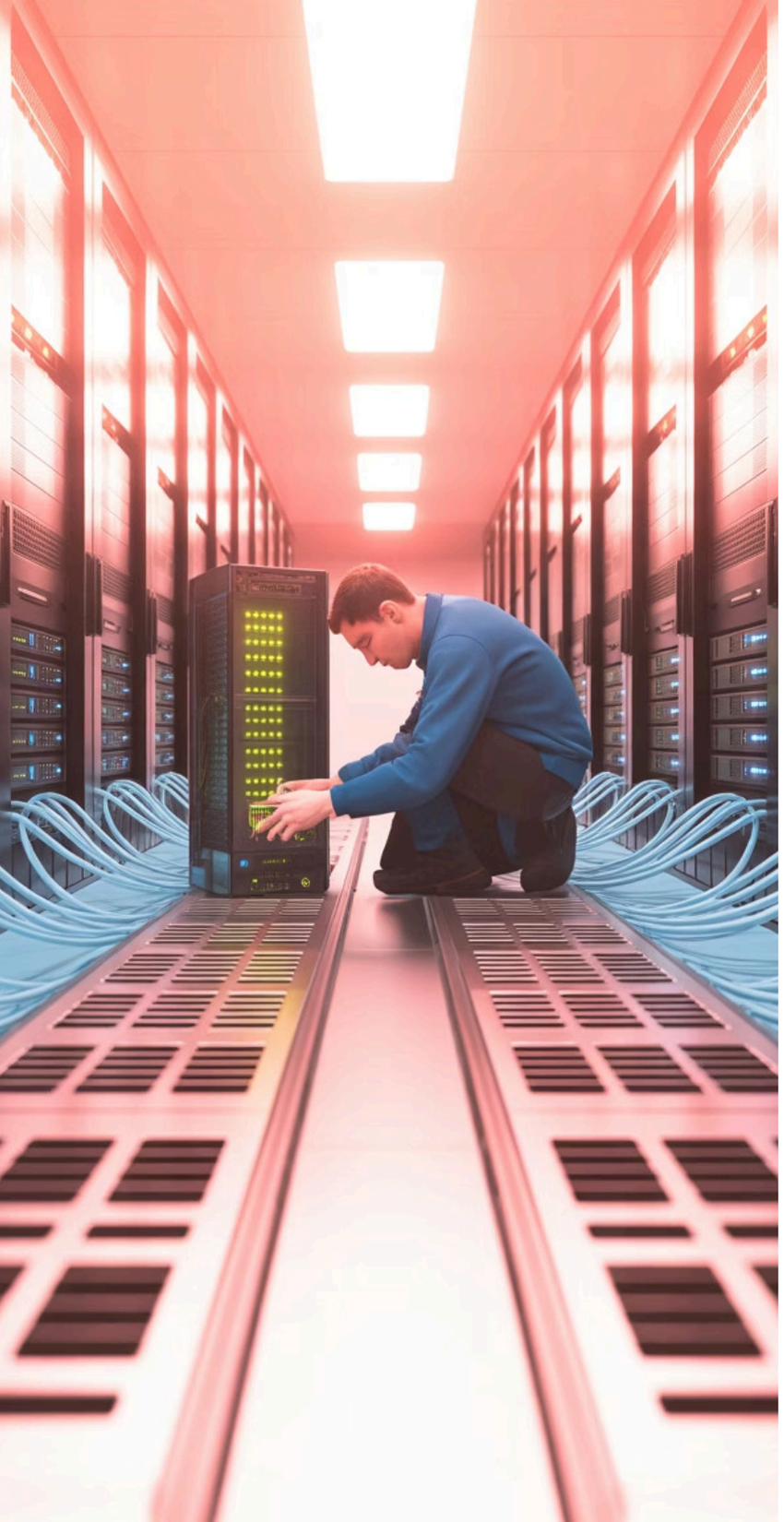
@Lazy Implementation

```
@Component
@Lazy
public class HeavyService {

    public HeavyService() {
        System.out.println("HeavyService Loaded!");
    }

    public void process() {
        // Heavy processing logic
    }
}
```

Without @Lazy, this class loads at application startup. With @Lazy, it loads only when the first method is called — saving both time and resources during the critical startup phase.

When to Use @Lazy

- Heavy initialization logic
- Database connections
- External service clients
- Large data processors
- Reporting services
- Optional features



Smart developers don't load everything at once — they load only what they need.

Learn Spring Best Practices at DURGASOFT

Why Choose DURGASOFT?

- Industry-recognized certification
- Real-world project experience
- Lifetime access to materials
- Placement assistance included
- Live doubt-clearing sessions
- Weekend batch availability

✨ *Join India's most trusted training institute for Spring Framework with SPRING BOOT Cloud.*

📞 9246212143

📞 8885252627

🌐 **www.durgasoftonline.com**

📩 **durgasoftonlinetraining@gmail.com**

DURGASOFT

Day-16: What Is @Qualifier in Spring and Why Do We Use It?

Master the @Qualifier annotation to resolve Bean ambiguity when multiple implementations exist

Resolving Bean Ambiguity with @Qualifier

The @Qualifier annotation is essential when multiple Beans of the same type exist in your application context and Spring doesn't automatically know which one to inject. By assigning each Bean a unique name or qualifier, you explicitly guide Spring to pick the correct implementation. This prevents confusion, avoids runtime errors, and ensures proper dependency wiring in real-world enterprise projects.

1 **Problem**

Multiple Beans of same interface type cause ambiguity during injection

2 **Solution**

Use @Qualifier to specify exact Bean name for injection

3 **Result**

Clear, predictable dependency resolution without errors

@Qualifier Usage Example

```
// Service Implementations
@Service("emailService")
public class EmailService implements MessageService {
    public void send(String message) {
        System.out.println("Email: " + message);
    }
}

@Service("smsService")
public class SmsService implements MessageService {
    public void send(String message) {
        System.out.println("SMS: " + message);
    }
}

// Injection with @Qualifier
@Autowired
@Qualifier("smsService")
private MessageService service;
```

Spring will inject **SmsService** specifically, not EmailService. This precise control is extremely valuable in enterprise applications where multiple implementations of the same interface exist for different business scenarios, protocols, or vendors.

EmailService

Sends notifications via email protocol

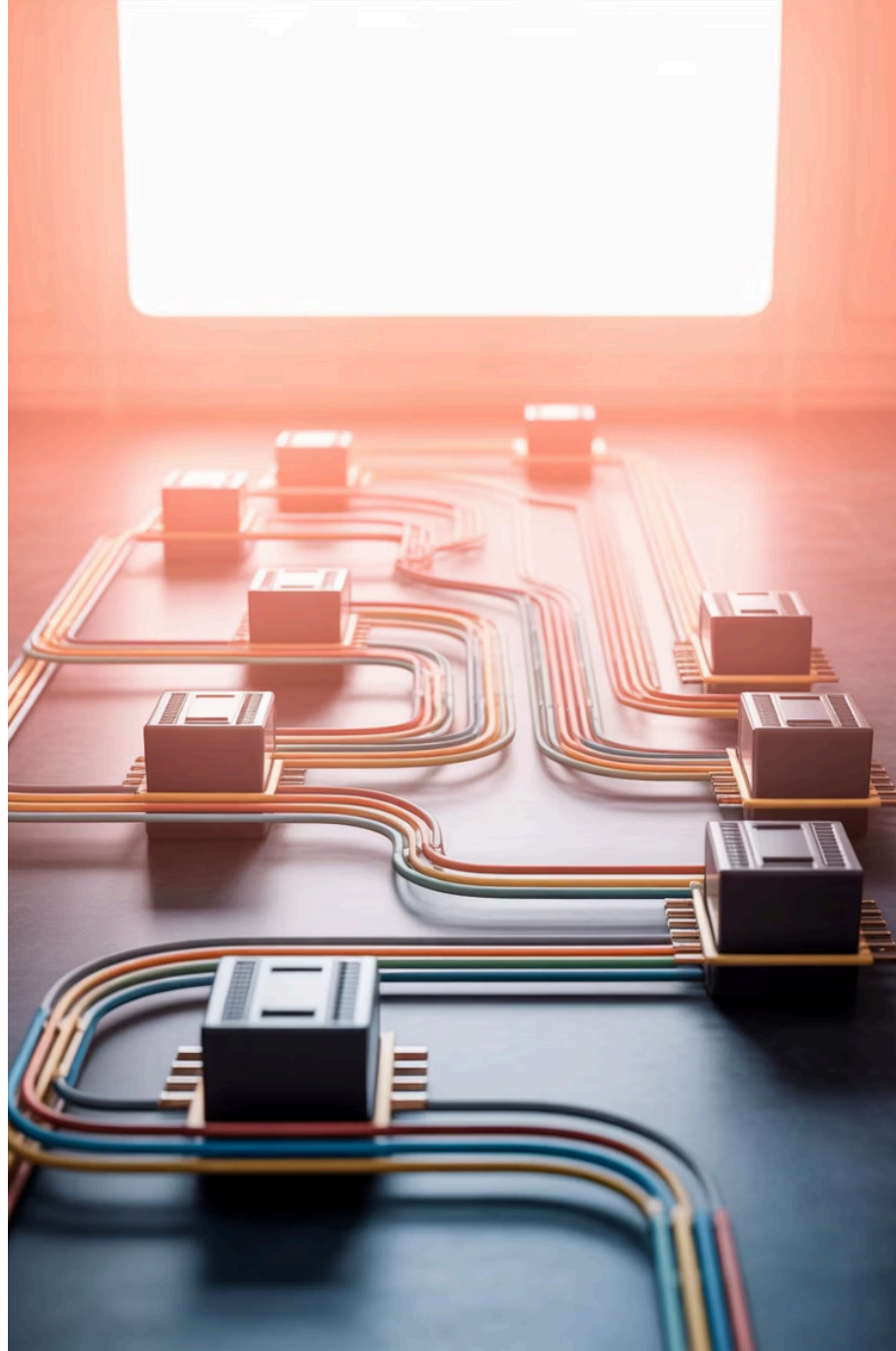
SmsService

Sends notifications via SMS gateway

PushService

Sends push notifications to mobile apps

Clarity in wiring brings clarity in your whole application.



Expert Spring Training at DURGASOFT

Certification

Industry-recognized certificate upon completion

Projects

Hands-on real-world project experience

Support

Lifetime doubt-clearing assistance

✨ *Transform your career with comprehensive Spring Framework and Spring Boot Cloud training from DURGASOFT – India's trusted training institute.*

📞 **Contact:** 9246212143, 8885252627 | 🌐 **www.durgasoftonline.com** | 📩 **durgasoftonlinetraining@gmail.com**

DURGASOFT

Day-17: What Is @Scope in Spring and When Should You Use It?

Explore different Bean scopes and learn when to apply each for optimal application behavior

Understanding Bean Scopes

The @Scope annotation defines how long a Bean should exist in the Spring container and how many instances Spring should create. Common scopes include **singleton**, **prototype**, **request**, **session**, and **application**. Selecting the appropriate scope ensures proper memory usage, thread safety, and predictable behavior in cloud and web applications.

Singleton (Default)

One instance per Spring container — shared across entire application

Prototype

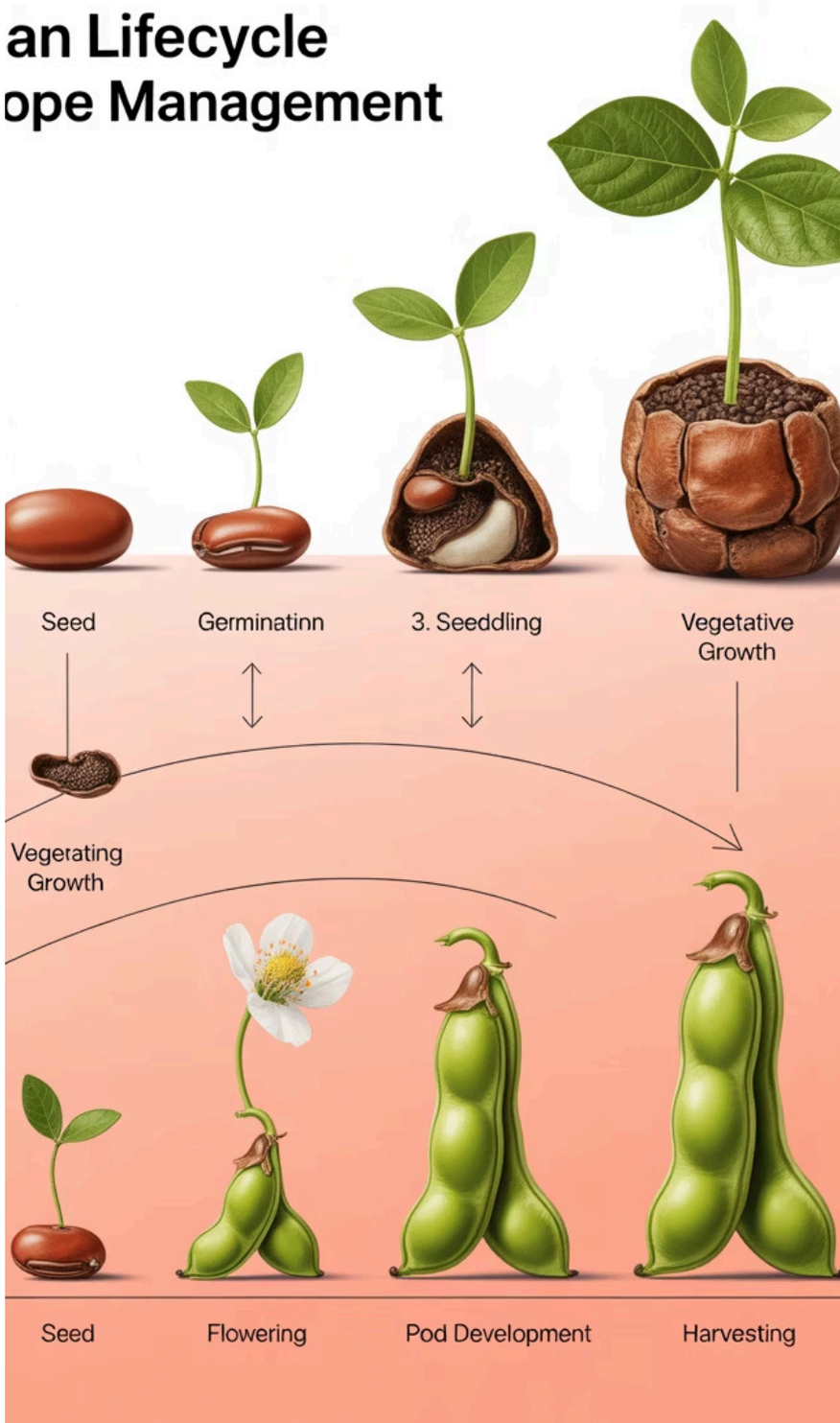
New instance created each time Bean is requested

Request

New instance per HTTP request (web applications only)

Session

One instance per HTTP session (web applications only)



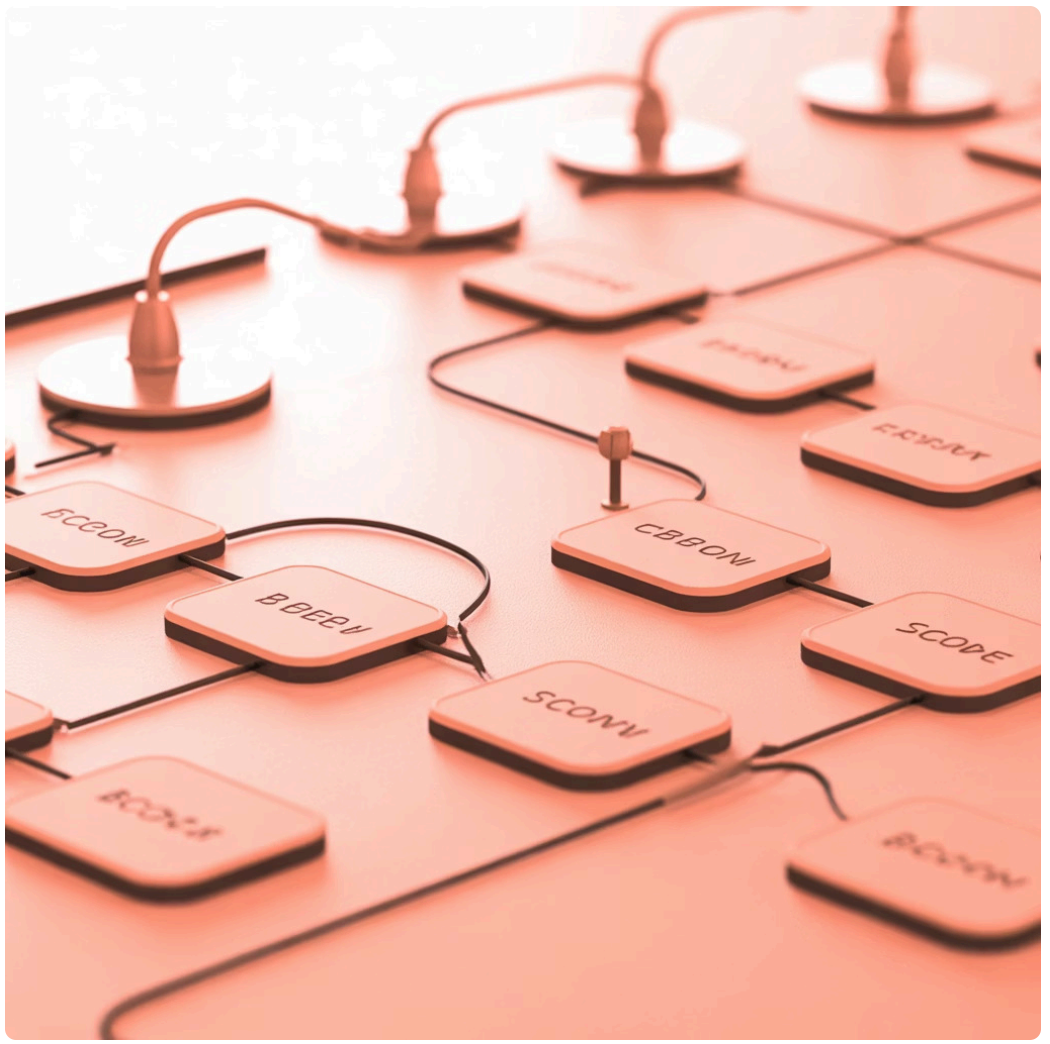
Prototype Scope Example

```
@Component
@Scope("prototype")
public class ReportGenerator {

    private String reportData;

    public void generateReport() {
        // Generate report logic
        reportData = fetchData();
    }
}
```

Here, Spring creates a **new object every time** someone requests the ReportGenerator Bean. This is particularly useful when the Bean holds temporary data, performs short-lived tasks, or requires independent state for each usage.

When to Use Each Scope:

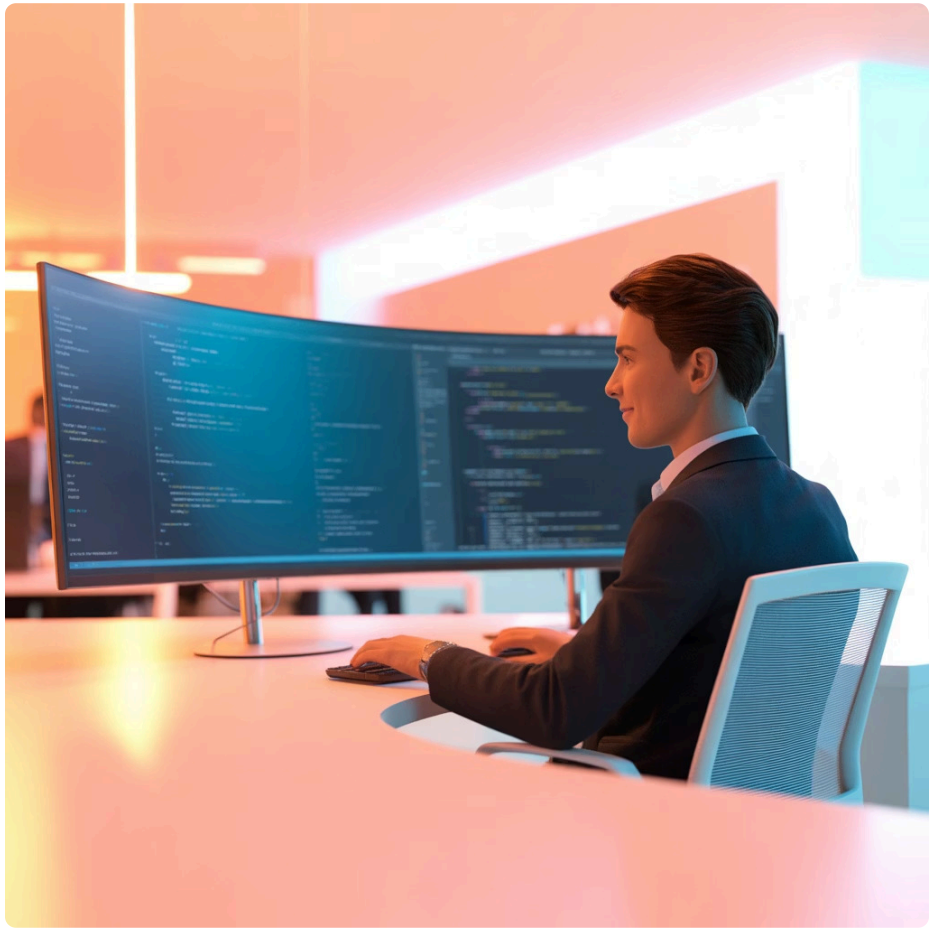
- **Singleton:** Stateless services, utilities, configuration
- **Prototype:** Stateful objects, temporary processing
- **Request:** Form backing objects, request-specific data
- **Session:** User session data, shopping carts



DURGASOFT

Right scope equals right performance. Small decisions make big improvements.

Elevate Your Skills with DURGASOFT



✨ Join DURGASOFT for comprehensive online training in SPRING Framework with SPRING BOOT Cloud – India's most trusted training institute for Java professionals.

Call Today

📞 9246212143, 8885252627

Visit Online

🌐 www.durgasoftonline.com

Email Us

📩 durgasoftonlinetraining@gmail.com

DURGASOFT

Day-18: What Is @Profile in Spring and Why Do We Use It?

Learn how @Profile enables environment-aware configuration for seamless multi-stage deployments

Environment-Specific Configuration

The @Profile annotation helps you load different Beans or configurations based on the active environment — such as **development**, **testing**, or **production**. This makes your application remarkably flexible and environment-aware, adjusting behavior automatically based on where it's deployed. This feature is extremely valuable in cloud deployments, CI/CD pipelines, and multi-stage applications where different configurations are required for each environment.

Development Profile

H2 database, debug logging, mock services, relaxed security for rapid development

Testing Profile

Test database, integration test configs, stubbed external services

Production Profile

Production database, error logging, real services, strict security measures

@Profile Configuration Example

```
@Configuration
@Profile("dev")
public class DevConfig {

    @Bean
    public DataSource dataSource() {
        HikariDataSource ds = new HikariDataSource();
        ds.setJdbcUrl("jdbc:h2:mem:testdb");
        return ds;
    }
}

@Configuration
@Profile("prod")
public class ProdConfig {

    @Bean
    public DataSource dataSource() {
        HikariDataSource ds = new HikariDataSource();
        ds.setJdbcUrl("jdbc:postgresql://prod-db:5432/app");
        return ds;
    }
}
```

If the application runs with the **dev** profile active, the DevConfig Bean loads. If the **prod** profile is active, ProdConfig loads instead. Spring automatically ignores Beans from inactive profiles. This approach keeps configurations clean, maintainable, and perfectly suited for each environment.

Smart developers don't change code for every environment — they use profiles.

Master Cloud-Ready Spring at DURGASOFT

Cloud-Native Focus

Learn Spring Boot Cloud for microservices and modern cloud deployments

Practical Training

Work on real enterprise projects with industry-standard practices

Career Support

Placement assistance and interview preparation included

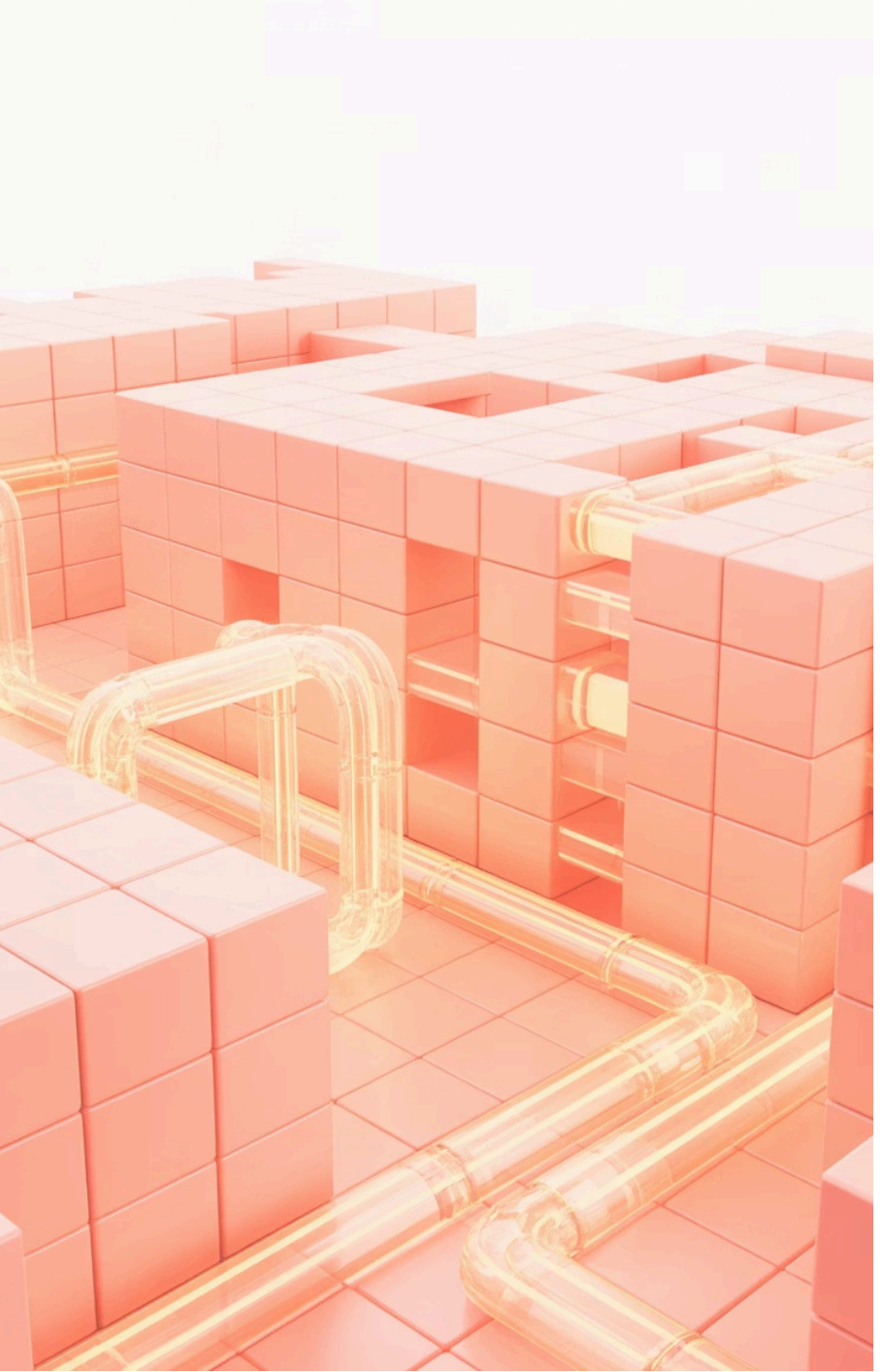
✨ *Join DURGASOFT – India's trusted training institute for comprehensive Spring Framework training.*

📞 **Contact:** 9246212143, 8885252627 | 🌐 **www.durgasoftonline.com** | 📩 **durgasoftonlinetraining@gmail.com**

DURGASOFT

Day-19: What Is @Import in Spring and When Should You Use It?

Discover how @Import helps organize complex configurations into clean, maintainable modules



Modular Configuration with @Import

The @Import annotation allows you to include one or more configuration classes inside another configuration class. This powerful feature helps you split large Spring projects into multiple clean, focused modules instead of cramming everything into a single massive configuration file. This modular approach significantly improves code readability, maintainability, and organization in enterprise applications.

Separate Concerns

Break configuration into logical modules (Security, Database, Messaging)

Import Modules

Use @Import to bring configurations together in main config class

Easy Management

Modify individual modules without affecting entire configuration

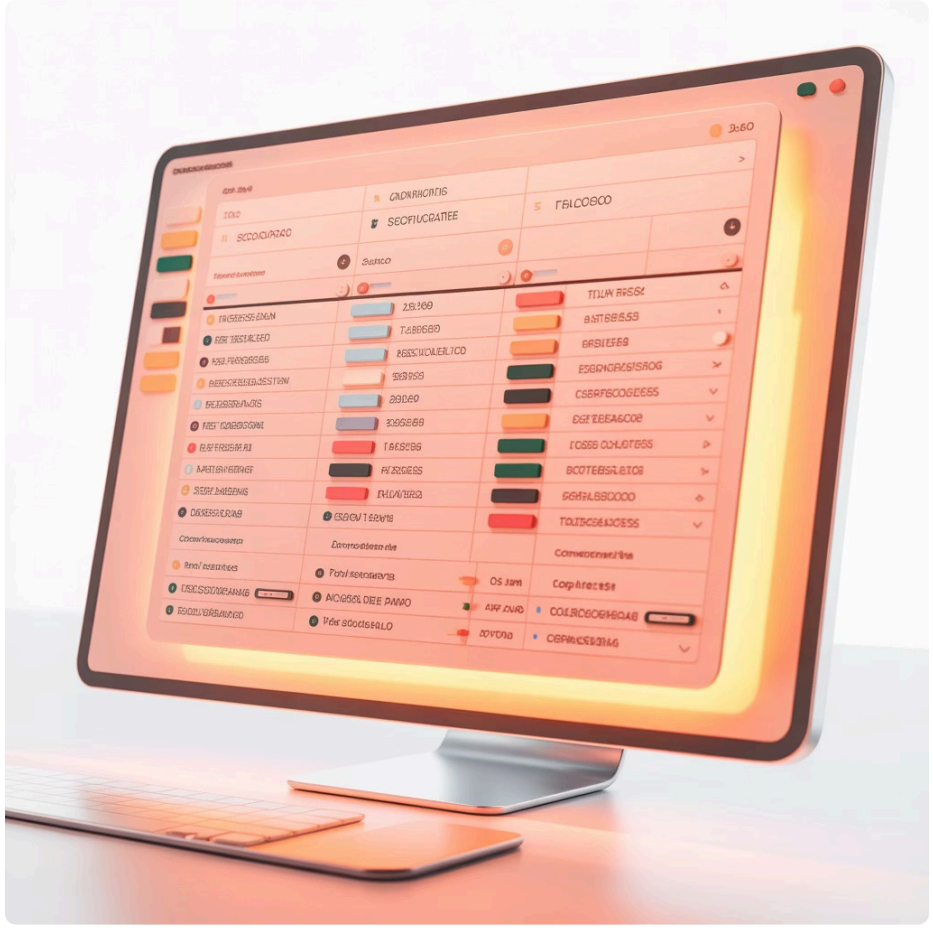
@Import Usage Example

```
@Configuration
@Import({
    SecurityConfig.class,
    DatabaseConfig.class,
    MessagingConfig.class
})
public class AppConfig {
    // Main configuration
}
```

Spring now loads SecurityConfig, DatabaseConfig, and MessagingConfig automatically when AppConfig is processed. This is extremely useful in large Spring Boot and cloud projects where configurations must be modular, testable, and maintainable across large development teams.

Benefits of Modular Config

- Better organization
- Easier testing
- Reusable modules
- Team collaboration
- Reduced complexity
- Clear separation



DURGASOFT

Good structure today saves countless hours tomorrow.

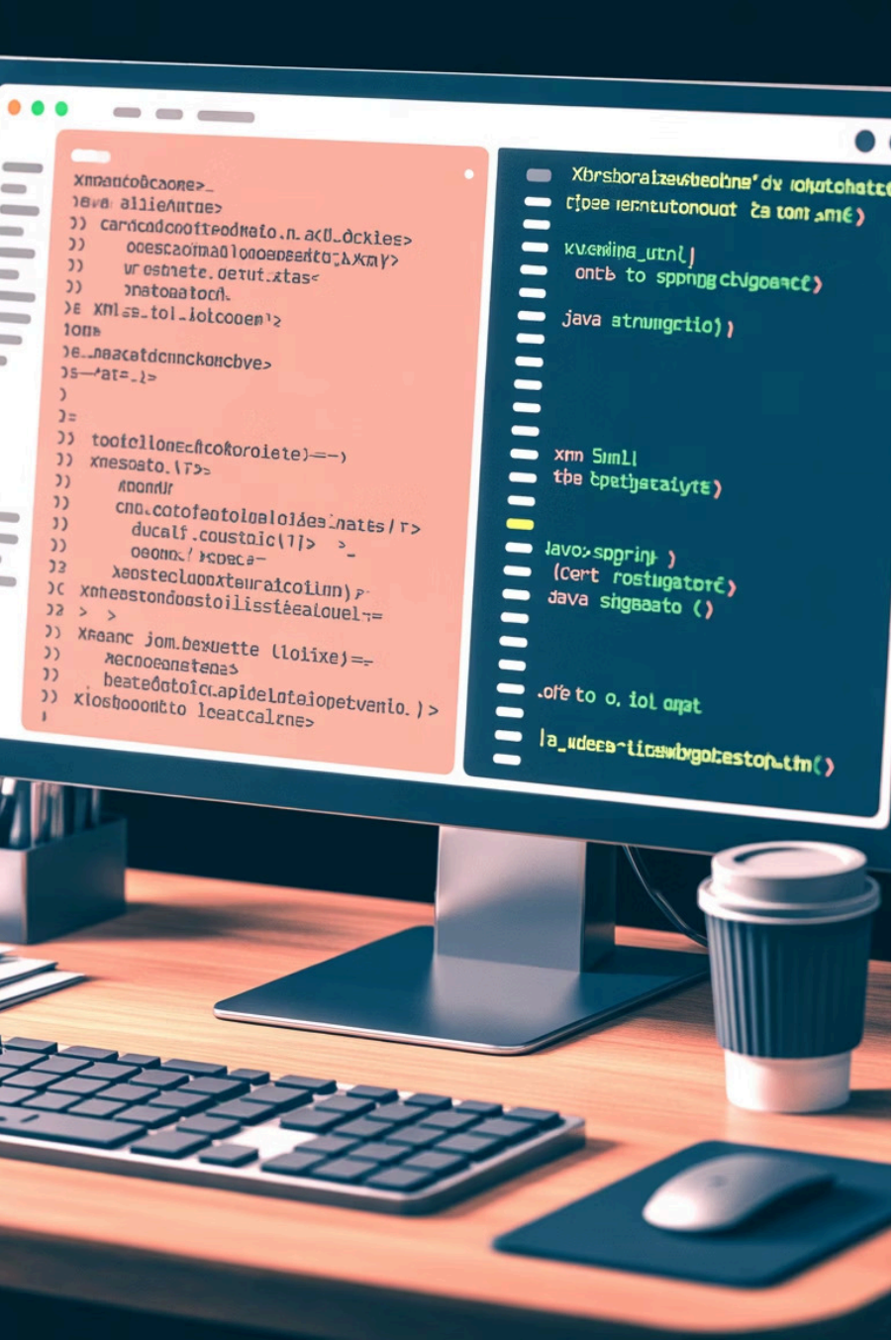
Build Enterprise Skills with DURGASOFT

What You'll Learn

- Spring Framework fundamentals
- Spring Boot best practices
- Microservices architecture
- Spring Cloud components
- Security and testing
- Production deployment

✨ *Join India's trusted training institute for Spring Framework with SPRING BOOT Cloud.*

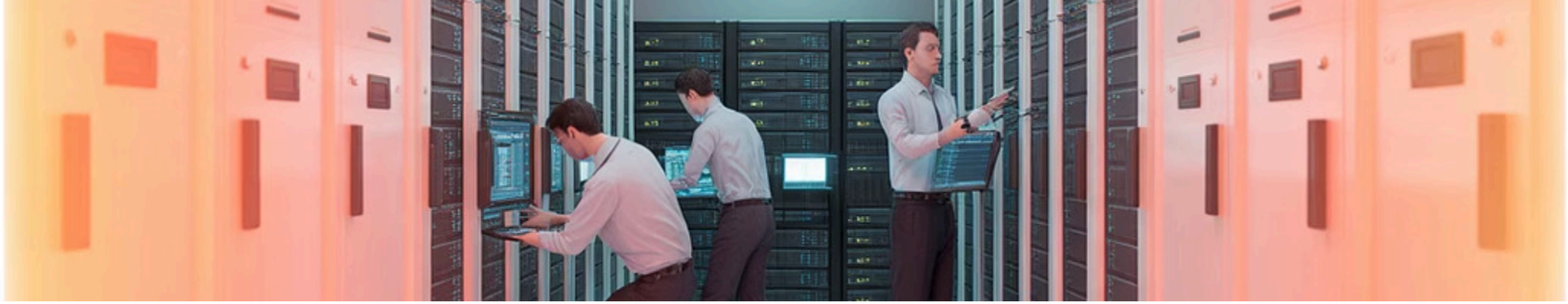
📞 9246212143, 8885252627 | 🌐 www.durgasoftonline.com | 📩 durgasoftonlinetraining@gmail.com



DURGASOFT

Day-20: What Is @ImportResource in Spring?

Learn how @ImportResource bridges legacy XML configuration with modern Java-based configuration



DURGASOFT

Bridging XML and Java Configuration

The @ImportResource annotation is used when you need to include an **existing XML configuration file** inside your Java-based Spring configuration. This is exceptionally helpful when migrating old Spring projects to modern Spring Boot applications, where you prefer Java configuration but still need to maintain some XML files for backward compatibility, third-party integrations, or gradual migration strategies.

Legacy XML

Old Spring projects used extensive XML configuration files

Modern Java

Gradually migrate to pure Java configuration approach

1

2

3

@ImportResource

Bridge annotation allowing XML and Java config coexistence

@ImportResource in Practice

Java Configuration

```
@Configuration
@ImportResource("classpath:applicationContext.xml")
public class AppConfig {

    @Bean
    public UserService userService() {
        return new UserService();
    }
}
```

Legacy XML File

```
<beans>
    <bean id="dataSource"
          class="org.apache.commons.dbcp.BasicDataSource">
        <property name="driverClassName"
                  value="com.mysql.jdbc.Driver"/>
        <property name="url"
                  value="jdbc:mysql://localhost/db"/>
    </bean>
</beans>
```

Spring will load all Bean definitions from applicationContext.xml along with your Java-based configurations. This is extremely useful during step-by-step migration from XML to annotations, allowing teams to modernize applications incrementally without massive rewrites.

Migration Strategy: Start by importing XML configs, then gradually convert each Bean to Java configuration. Test thoroughly after each conversion step to ensure nothing breaks.

Migration becomes easy when you mix the old and the new wisely.

Start Your Spring Journey with DURGASOFT Today

Comprehensive Training Program

Complete Spring Framework and Spring Boot Cloud training covering all essential annotations, configurations, and best practices

Expert Guidance

Learn from experienced professionals with years of industry experience in enterprise Spring development

Career Advancement

Get placement assistance, interview preparation, and industry-recognized certification to boost your career

✨ *Join DURGASOFT – India's most trusted training institute for Spring Framework with SPRING BOOT Cloud training. Transform your career with expert-led, practical training.*

10K+	95%	15+
Students Trained	**Placement Success**	**Years Experience**

📞 **Call Us:**
9246212143
8885252627

🌐 **Visit:**
www.durgasoftonline.com

📩 **Email:**
durgasoftonlinetraining@gmail.com