Java-Based Configuration

Java-based configuration in Spring allows us to configure beans using Java classes instead of traditional XML configuration.

Steps:

1. Create a class annotated with @Configuration. This class will contain methods that return instances of the beans you need.
2. Use AnnotationConfigApplicationContext to initialize the Spring container and provide it with the configuration class.
3. Retrieve beans from the container using context.getBean().

Configuration Class:

```
import org.springframework.context.ApplicationContext;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

// Configuration class
@Configuration
public class AppConfig {
        @Bean
        public Desktop desktop() {
        return new Desktop();
        }
}
```

Bean Class:

```
// Bean class
class Desktop {
        public Desktop() {
      System.out.println("Desktop Object Created");
        }

        public void compile() {
      System.out.println("Compiling using Desktop");
        }
}
```

Main Class:

```
// Main class to test the configuration
public class App {
        public static void main(String[] args) {
        ApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);

        // Retrieve the Desktop bean from the context
        Desktop dt = context.getBean(Desktop.class);
      dt.compile();  // Output: "Compiling using Desktop"
        }
}
```

Key Points:

- **@Configuration**: Marks the class as a source of bean definitions.
- **@Bean**: Defines a bean, and the method name (desktop) becomes the bean name by default.
- **AnnotationConfigApplicationContext**: Initializes the Spring context with the provided configuration class.