AOP Notes

Spring Core

By MKT

How to Create bean

1.Using Xml configuration

```
<bean id="myService" class="com.example.MyService"/>
```

2.Using @component configuration

```
@component
Public class MyService{}
```

3.Using @Bean in a @configuration class

```
class AppConfig{
   @bean
   Public void myService(){}
}
```

Types of DI in spring

1.Constructor Injection

```
public class MyClient {
    private final MyService service;

    public MyClient(MyService service) {
        this.service = service;
    }
}
```

Types of DI in spring

2.Spring bean configuration

```
@Bean
public MyClient myClient(MyService service) {
    return new MyClient(service);
}
```

Types of DI in spring

3.Setter Injection

```
public class MyClient {
    private MyService service;

    public void setService(MyService service) {
        this.service = service;
    }
}
```

Types of DI in spring

4.Field Injection

```
public class MyClient {
    @Autowired
    private MyService service;
}
```

Key Annotation in spring core

Annotation	**Purpose**
\`@Component\`	Marks a class as a Spring-managed bean.
\`@Service\`, \`@Repository\`, \`@Controller\`	Specializations of \`@Component\`.
\`@Autowired\`	Tells Spring to inject a dependency.
\`@Qualifier\`	Resolves ambiguity when multiple beans of same type exist.
\`@Bean\`	Declares a bean in Java config (\`@Configuration\` class).
\`@Configuration\`	Marks a class that declares \`@Bean\` methods.

Bean Scope

Scope	**Description**
`singleton` *(default)*	Only one instance per Spring container.
`prototype`	New instance every time it's requested.
`request`	One instance per HTTP request (Web context).
`session`	One instance per HTTP session (Web context).

```
@Component
@Scope("prototype")
public class MyPrototypeBean { }
```

@Primary Vs @Qualifier

When multiple beans of the same type exist, Spring doesn't know which one to inject.

```
@Component
@Primary ( default bean)
public class DefaultService implements MyService { }
```

```
@Component(“customService")
public class CustomService implements MyService { }
```

```
@Autowired
@Qualifier("customService") (specific bean)
private MyService service;
```

@Lazy Vs @Eager

By default, Spring creates **singleton beans eagerly** during startup.

- `@Lazy`: Bean is initialized **on first use**.
- `@Lazy(true)` can also be used in XML or config classes.

Common Pointcut Designators in AOP

Designator	Purpose	Matches...
`execution`	Match method execution	Method signature (interface or class method execution)
`within`	Match join points within a specific type or package	Class or package scope
`this`	Match based on the **proxy object's** type	The proxy/interface type (especially in Spring AOP)
`target`	Match based on the **actual target object's** type	The runtime type of the proxied object
`args`	Match based on the method's argument types	Runtime argument types passed to the method
`bean`	Match by Spring **bean name**	The bean ID from Spring context
`@target`	Match if the **target object’s class** has a specific annotation	Classes annotated with a given annotation
`@within`	Match join points in types with a specific annotation	Classes annotated with a given annotation
`@annotation`	Match method with a specific annotation	Methods annotated with a specific annotation
`@args`	Match if **arguments** are annotated with a specific annotation	Arguments of method annotated with specific annotation

Summary Table

Use When You Want To...	**Use This Designator**
Match methods based on signature	\`execution\`
Match methods inside a class or package	\`within\`
Match proxy type	\`this\`
Match actual object type	\`target\`
Match methods by parameter types at runtime	\`args\`
Match specific Spring bean by name	\`bean\`
Match methods or classes with annotations	\`@annotation\`, \`@within\`, \`@target\`
Match if method arguments are annotated	\`@args\`