Developed by Rod Johnson in 2003

Date.
Page No.

Spring Framework

⇒ Spring is a Dependency Injection framework to make Java application loosely coupled.

⇒ Powerful Lightweight application development framework.

⇒ Makes easy development of JavaEE application.

⇒ Simplicity, Testability and loose coupling.

Feature of Spring :-

1. Open Source
2. Comprehensive tool
3. Light weight
4. Solves Problems
5. Framework of framework
6. Avails array of resources.

Simplicity

1. Spring is simple because it is non-invasive. It uses POJO and POJI

2. If a Java class is not coupled with any technology or any framework then that java class is called POJO (Plain old java object). If a java interface is not coupled with any technology or any framework then that java interface is called POJI (Plain Old Java Interface)

Date
Page No.

Testability

1. For writing the spring application server [container] is not mandatory because it has its own container to run the application.

2. Spring framework already have testing layers available. ex. JUnit already integrated.

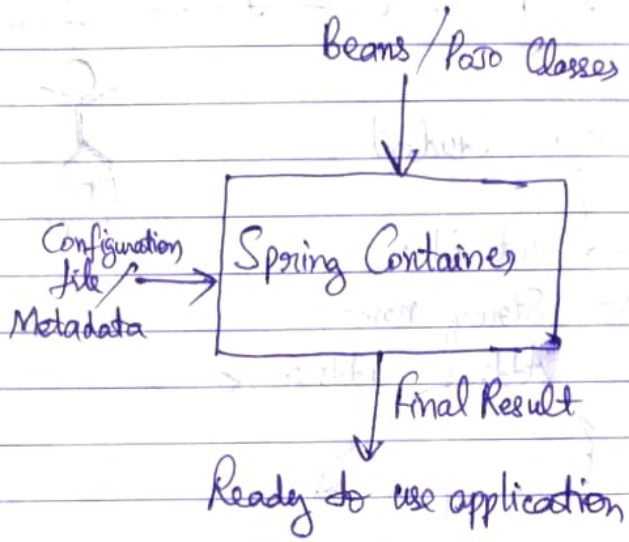
POM: Project Object Model

Date.
Page No.

Spring IOC container

↳ Creating the object
↳ Holding the object in the memory
↳ Injecting one object into another (Dependency Injection)
↳ Maintain lifecycle of object.

Spring IOC containes by using Java POJO classes and configuration metadata produces a fully configured and executable system or application.



Spring containes will be responsible to construct the Java objects by parsing the XML files.

Types of IOC containers

Bean factory

* constructs the object when we request for it by calling getBean method
* simplest containes providing the basic support for dependency injection.

ApplicationContext

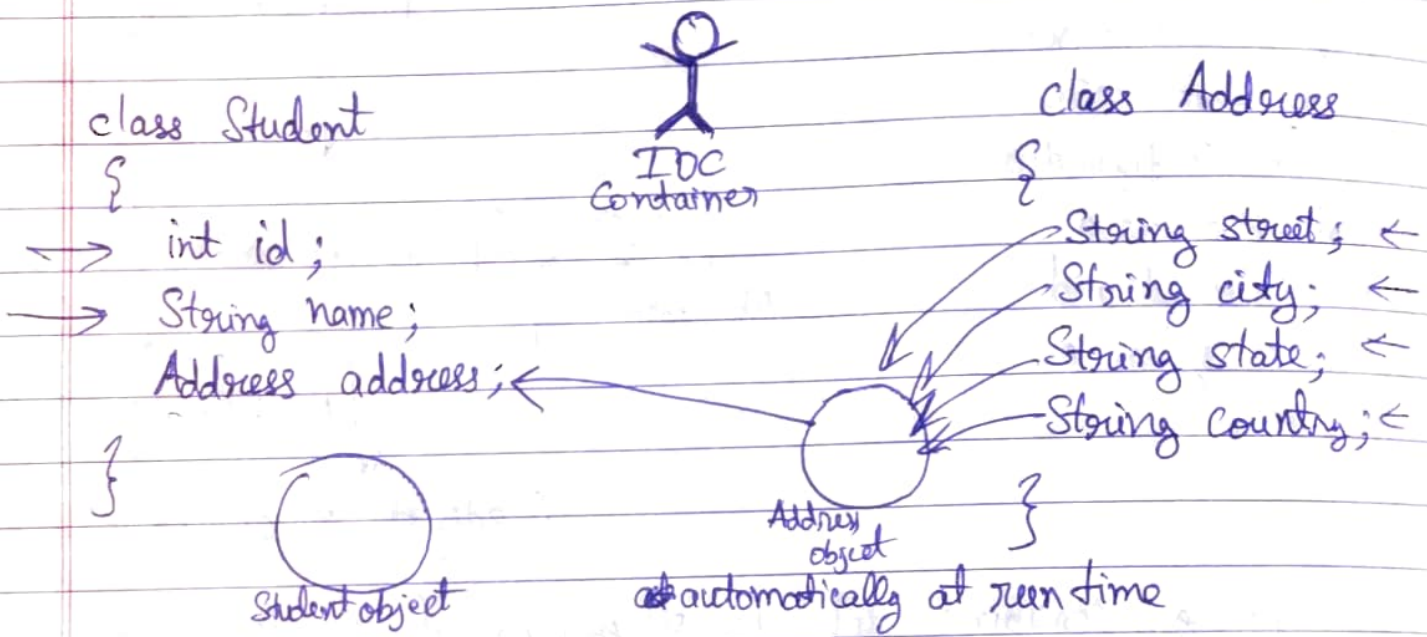
* Constructs the object even though we dont request for it.
* Built on top of bean factory interface. This containes adds more enterprise specific functional

Date
Page No.

⇒ Application Context

- → ClasspathXMLApplicationContext
- → AnnotationConfigApplicationContext
- → FileSystemXMLApplicationContext

⇒ Dependency Injection (DI) is a design pattern



Student class depending on Address class.

Date
Page No.

⇒ Dependency Injection can be done in 2 ways

Setter Injection

<property> subelement of <bean> is used for setter injection

Constructor Injection

<constructor-arg> subelement of <bean> is used for constructor injection.

⇒ Data types (Dependencies)

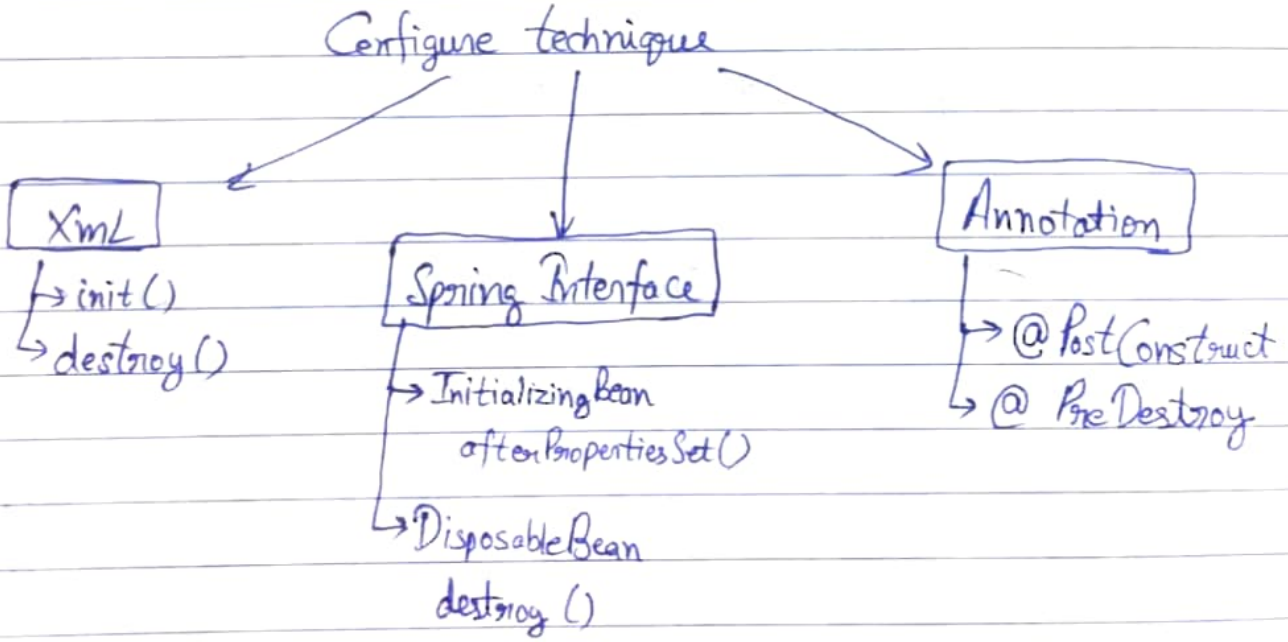
1. Primitive Data Types: byte, short, char, int, float, double, long, boolean
2. Collection Types: List, Set, Map and Properties
3. Reference Type (User defined): Other class object.

⇒ Ambiguity Problem and its solution with constructor injection.

Date.
Page No.

⇒ Life cycle Methods

object created
↓
Setting Value Properties
↓
init ()
↓
Then we read and use the bean
↓
destroy ()
↓
object destroyed



Date.
Page No.

⇒ Autowiring in Spring

* Feature of spring framework in which spring container inject the dependencies automatically.

* Autowiring can't be used to inject primitive and string values. It works with reference only.

Autowiring
- xml
  - Autowiring Modes
    - default: no
    - by Name
    - by Type
    - Constructor
    - auto detect — deprecated since Spring 3
- Annotation
  - @Autowired
    - on property
    - on setter
    - on constructor

uses byType

Advantages :-
→ Automatic
→ less code

Disadvantages :-
→ No control of programmer.
→ It can't be used for primitive and string values.

@Qualifier can be used above property and setter method and not above constructor

Date
Page No.

⇒ @Qualifier annotation with Autowiring

```
@Autowired
@Qualifier(" ")
```

When we have multiple beans in configuration file, then we put name of the bean to be picked by using Qualifier annotation.

⇒ Spring Standalone Collections.

```
<util:list  list-class="java.util.LinkedList" id=" ... " />
<util:map  map-class="java.util.HashMap" id=" " />
<util:properties  id=" " />
```

⇒ Stereotype Annotations

```
@Component
class Student
{

}

Student student = new Student();
```

```
@Component("ob")
class Student
{

}

ob
```

```
<context:component-scan base-package=" " />

@Value(" ")
```

Date
Page No.

⇒ Collection with @Value

```
@Value("#{temp}")
private List<String> address;
```

Standalone collection

```
<util:list list-class="java.util.ArrayList" id="temp" />
<value>Delhi</value>
<value>Noida</value>
<value>Dehradun</value>
```

⇒ Spring Bean Scope

singleton
prototype
request
session
globalsession — for portlet applications.

{ Web application }

(by default) singleton → Only single object created by spring container and provided to you everytime.

prototype → everytime we call, spring container will create new object of the bean

Configure bean scope

- XML: `<bean class="" name="" scope="" />`
- Annotation: `@Component` `@Scope("")`

Date
Page No.

⇒ Spring Expression Language (SpEL)

Supports parsing & executing expressions with the help of @Value annotations.

```
@Value("#{Expression}")
```

expression → Classes, Variable, Methods, Constructors & Objects and symbols.
char, numerics, operators, keywords and special symbols which return a value.

eg. `@Value("#{11+22}")`

`@Value("#{8>6 ? 88 : 55}")`

* Invoking static method and variable

```
T(class).method(param)
T(class).Variable
```

```
@Value("#{T(java.lang.Math).sqrt(144)}")
@Value("#{T(java.lang.Math).E}")
```

* Boolean type with SpEL

`@Value("#{8>3}")`

Date
Page No.

⇒ Removing Complete XML for Spring Configuration.

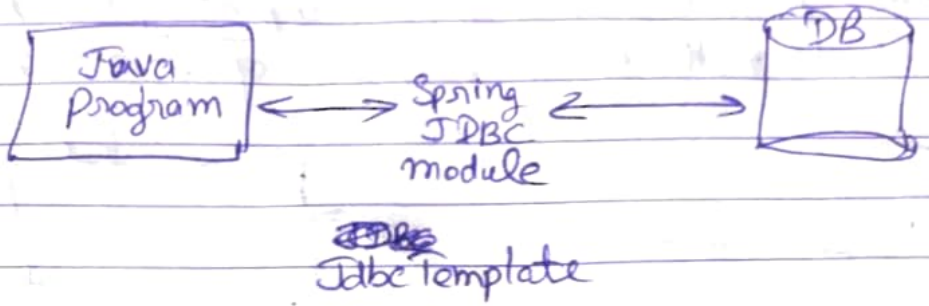
```
ApplicationContext context =
        new AnnotationConfigApplicationContext( .class);

@Configuration
@ComponentScan(basePackages = "")
@Bean
```

Airflow

Date
Page No.

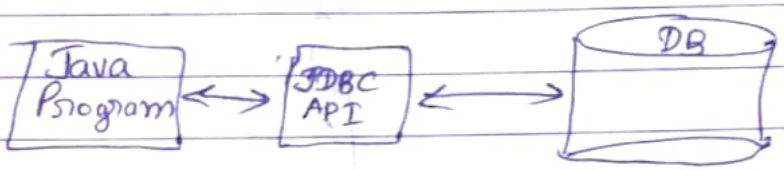
Spring JDBC

Spring JDBC is a powerful mechanism to connect to the database and execute SQL queries.



⇒ JDBC concept

JDBC is API to perform operation with database.



⇒ Problems of JDBC

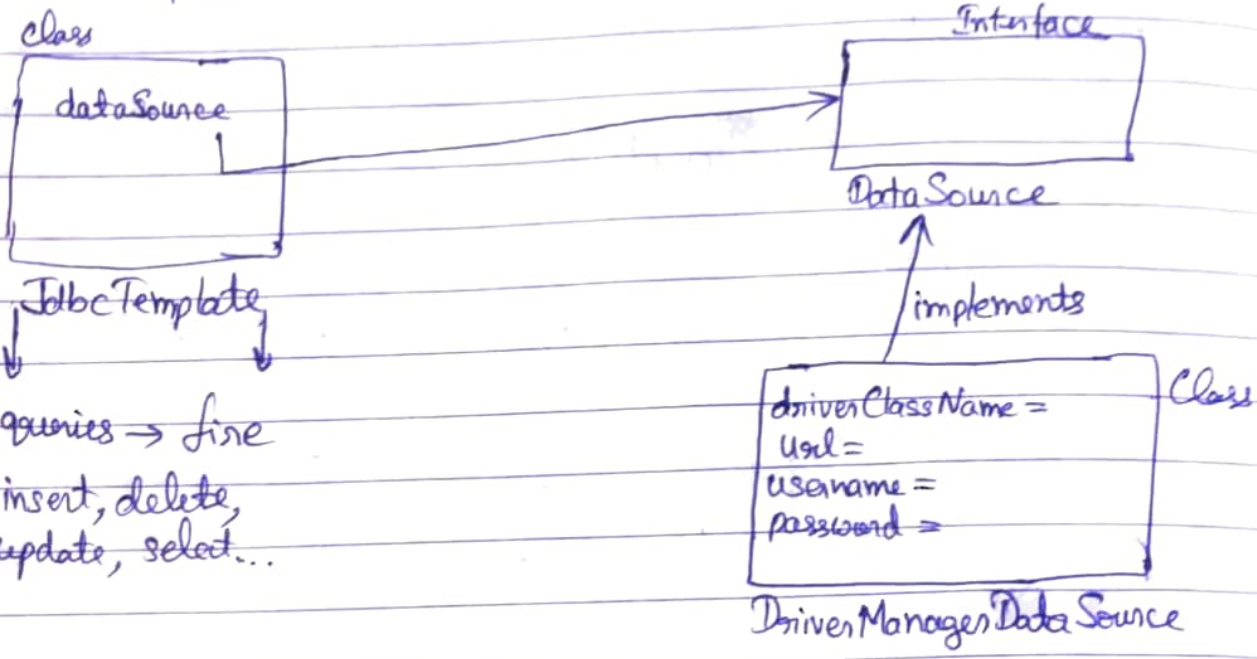
* We need to write a lot of code
  ↳ Connection Open, statement, execute, connection close.
* Exception Handling Problem : checked Exception
  SQL Exception
* Repeating of all these codes from one to another database logic is a time consuming task.

Date
Page No.

Advantage of Spring JDBC

* Solution of JDBC problems are provided by Spring JDBC

* Spring JDBC provide class **JdbcTemplate** which has all the important methods to perform operation with database.



JdbcTemplate methods :-

update () → insert, update, delete

execute () → select queries.