

# Spring Framework

Dependency Injection



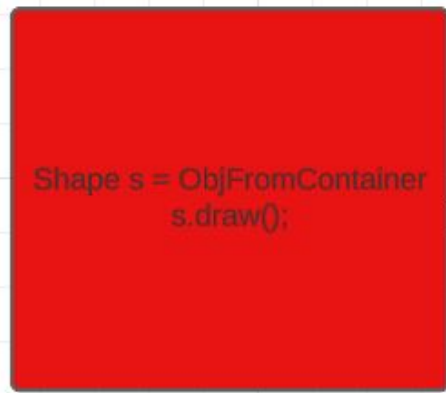
# Table of Content

- Dependency Injection
- Old Style for using object from another class
- Injection Types
- Constructor Injection
- Constructor Injection dev process(Step by Step)
- Time to Write code
- Break
- Setter Injection
- Setter Injection dev process(Step by Step)
- Questions
- Thanks

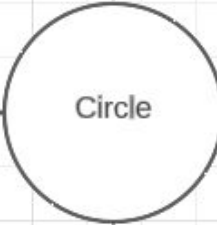
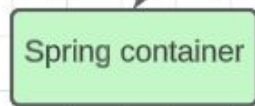
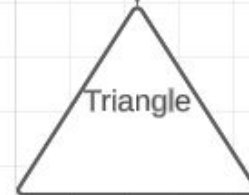
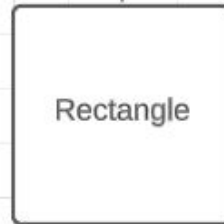
# Dependency Injection

is a technique where the dependencies of a class (i.e., the objects it relies on) are injected into the class rather than the class creating or managing its dependencies





config file which contains the beans definition



give me a Shape

give me an object

Object



**Let's see the old  
style code**



# Injection Types

We will cover the two most common (without annotations)

- Constructor Injection
- Setter Injection

We will talk about “auto-wiring” in the Annotations section later



# Injection Types (differences)

Feature	Setter Injection	Constructor Injection
<b>Flexibility</b>	More flexible, especially for optional dependencies.	Dependencies are set at the time of object construction.
<b>Dependency Overriding</b>	Easier to override dependencies by calling setters.	Dependencies are typically set once and not changed.
<b>Initialization Order</b>	Initialization may be partial until all setters are called.	Initialization is complete once the constructor finishes.
<b>Guarantees Required Dependencies</b>	Does not guarantee that required dependencies are set.	Ensures that all required dependencies are provided during construction.



**Let's see the old code of  
constructor injection**





# Dev Process - Constructor Injection

1. Define the dependency class
2. Create a constructor in your class for injections
3. Configure the dependency injection in Spring config file



# Step 1: Define the dependency class

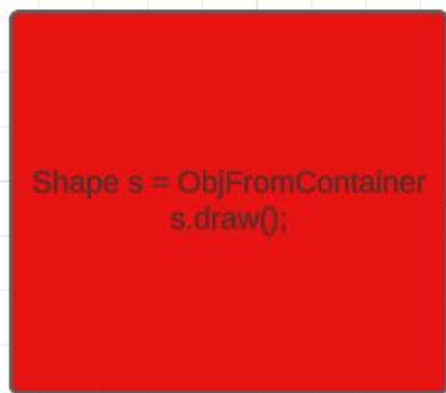
```
3 public class DrawTool
4 {
5     1 usage
6     public void draw(String shapeName) { System.out.println("Drawing a 2d for " + shapeName); }
7 }
8 }
```



## Step 2: Create a constructor in your class for injections

```
public class CircleShape implements Shape
{
    //this is the object that we injected it inside the circle class
    public DrawTool drawTool;

    //this is the constructor injection
    public CircleShape(DrawTool drawTool) {
        this.drawTool = drawTool;
    }
}
```



give me a Shape

Spring Bean  
Container

config file

config file which  
contains the beans  
defination

Shape

Circle

Rectangle

Triangle

DrawShape

Spring container

give me an object

Object

```
<!--
```

```
<bean
```

```
</bean
```

```
<!--
```

```
<bean
```

```
</bean
```



**Let's write some code**



**5 minutes Break**



**Let's see the old code of Setter injection**



# Dev Process - Setter Injection

1. Define the dependency class
2. Create a Setter method in your class for injections
3. Configure the dependency injection in Spring config file





# Step 1: Define the dependency class

```
3 public class DrawTool
4 {
5     1 usage
6     public void draw(String shapeName) { System.out.println("Drawing a 2d for " + shapeName); }
7 }
8 }
```



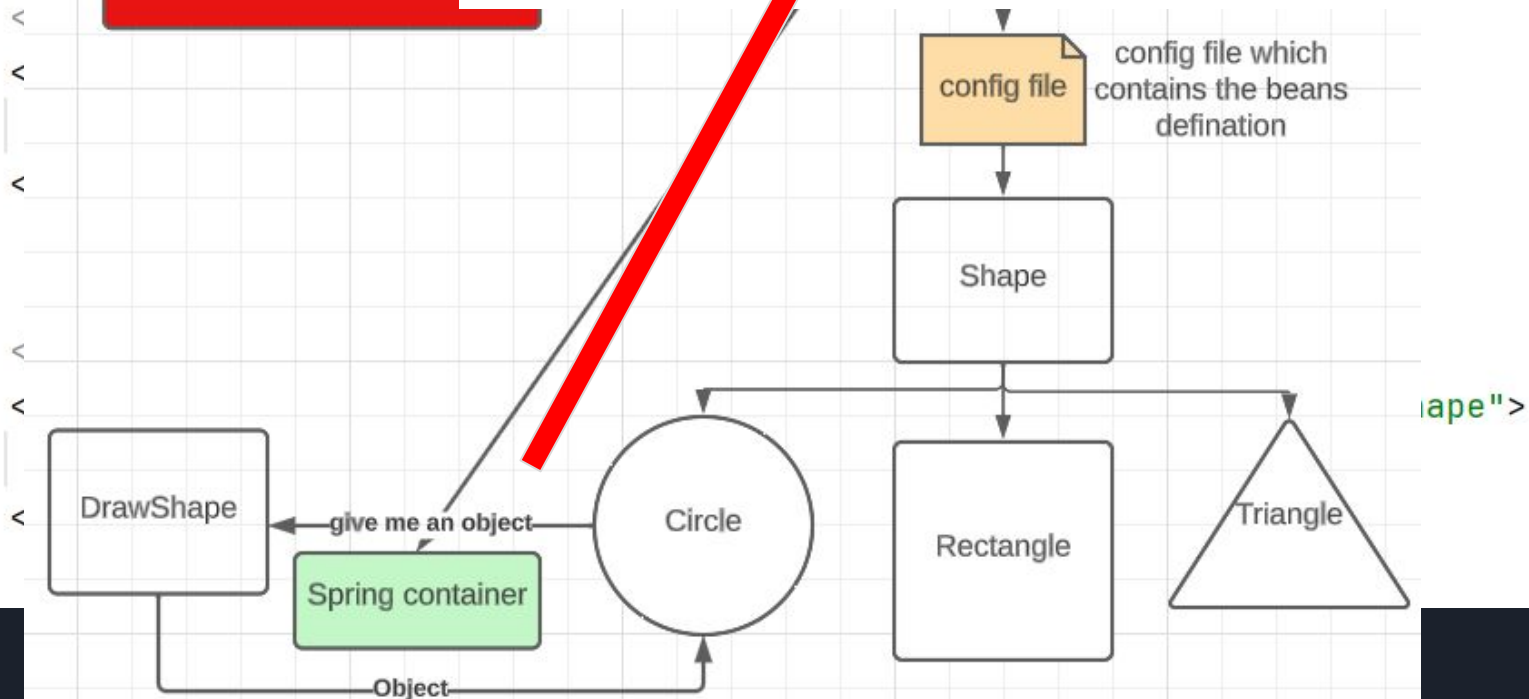
## Step 2: Create a constructor in your class for injections

```
public class RectangleShape implements Shape
{
    public Draw2D drawShape;

    //setter method for setter injection
    public void setDrawShape(Draw2D drawShape) {
        this.drawShape = drawShape;
    }
}
```

```
Shape s = ObjFromCont  
s.draw();
```

```
//setter method for setter injection  
public void setDrawShape(Draw2D drawShape) {  
    this.drawShape = drawShape;  
}
```



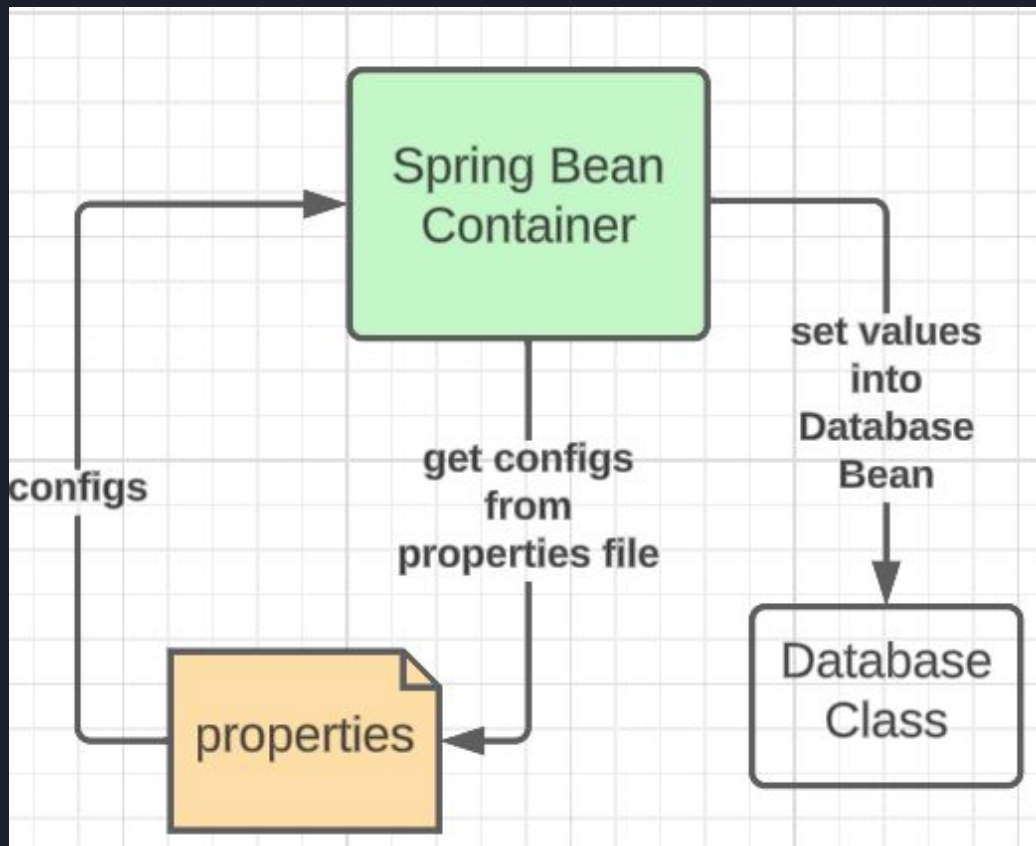


# Injecting Values from a properties file

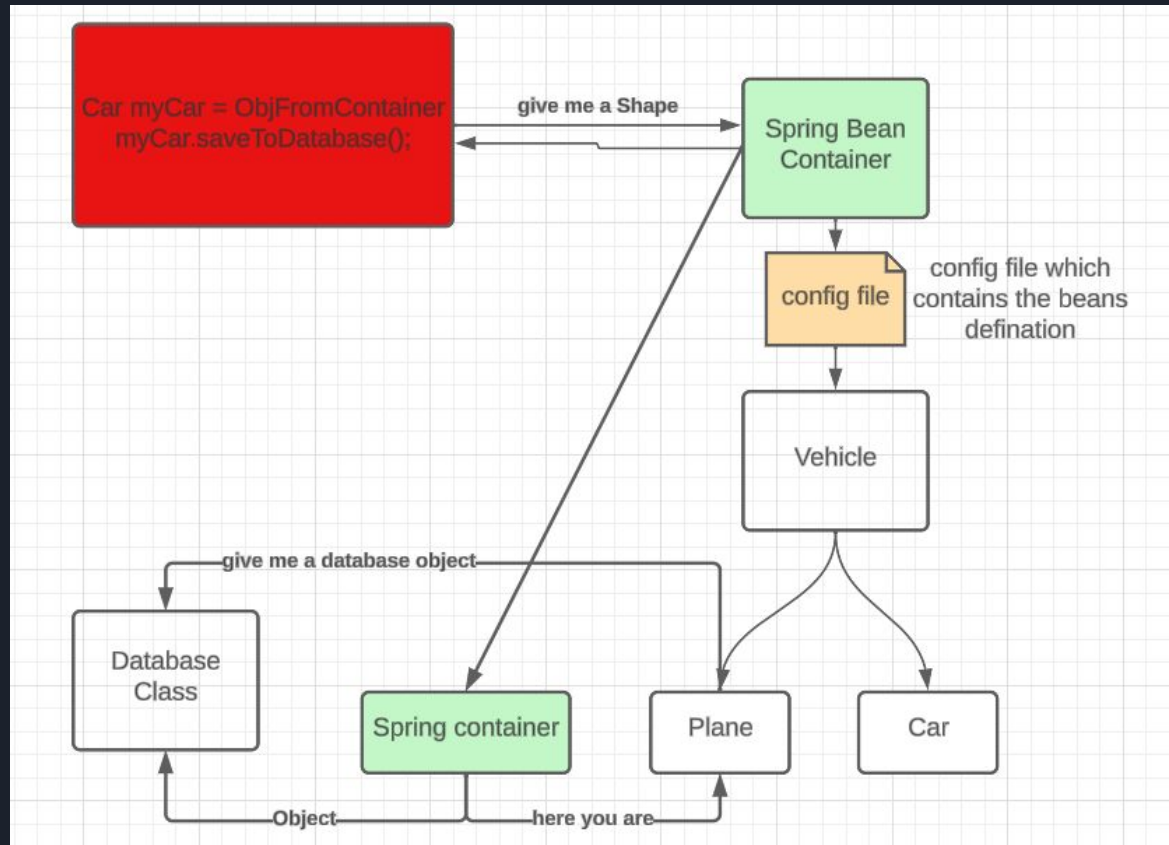
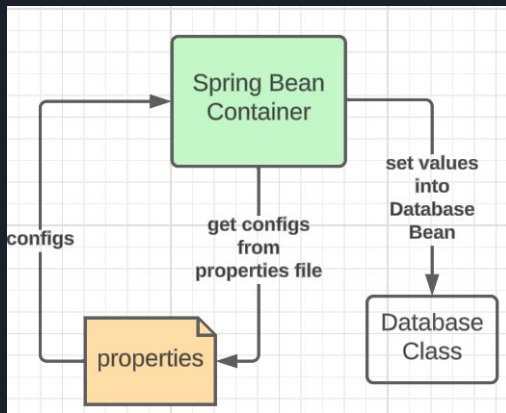


## Use Case

We have a backend system that store some vehicles with two types (car , plane) . we need to store the brand of each vehicle into the database. You will take the database properties details (**url , username , password**) and store them into a properties file.



The container will get the properties and then inject them to the database bean



Now , the container will create a database object for the plane to store its data into the database



# Development Steps

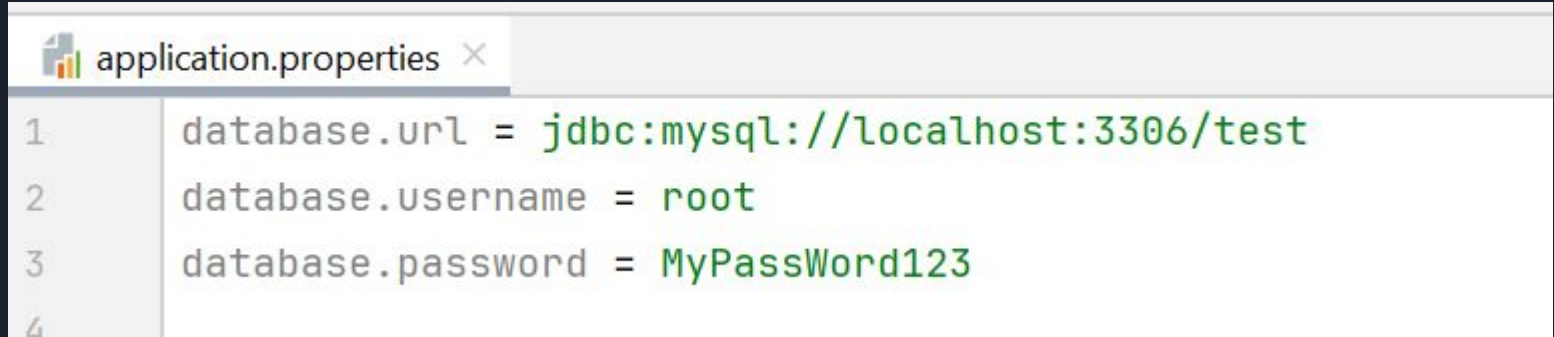
Step 1: Create Properties File

Step 2: Load Properties file in Spring config file

Step 3: Reference Values from Properties File




# Step 1: Create Properties File



The screenshot shows a code editor window with a single tab titled 'application.properties'. The editor contains three lines of text, each representing a database connection property. The first line is 'database.url = jdbc:mysql://localhost:3306/test', the second is 'database.username = root', and the third is 'database.password = MyPassWord123'. The text is color-coded: 'jdbc:mysql' is green, 'localhost:3306/test' is dark blue, 'root' is red, and 'MyPassWord123' is green. Line numbers 1, 2, 3, and 4 are visible on the left side of the editor.

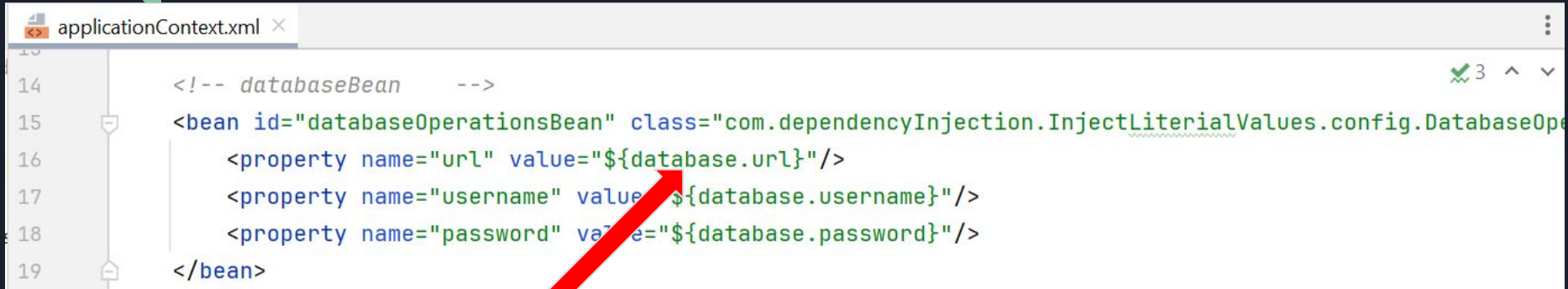
```
1 database.url = jdbc:mysql://localhost:3306/test
2 database.username = root
3 database.password = MyPassWord123
4
```

## Step 2: Load Properties file in Spring config file



```
applicationContext.xml x
9
10    <!-- load the properties file    -->
11    <context:property-placeholder location="classpath:application.properties"/>
12
```

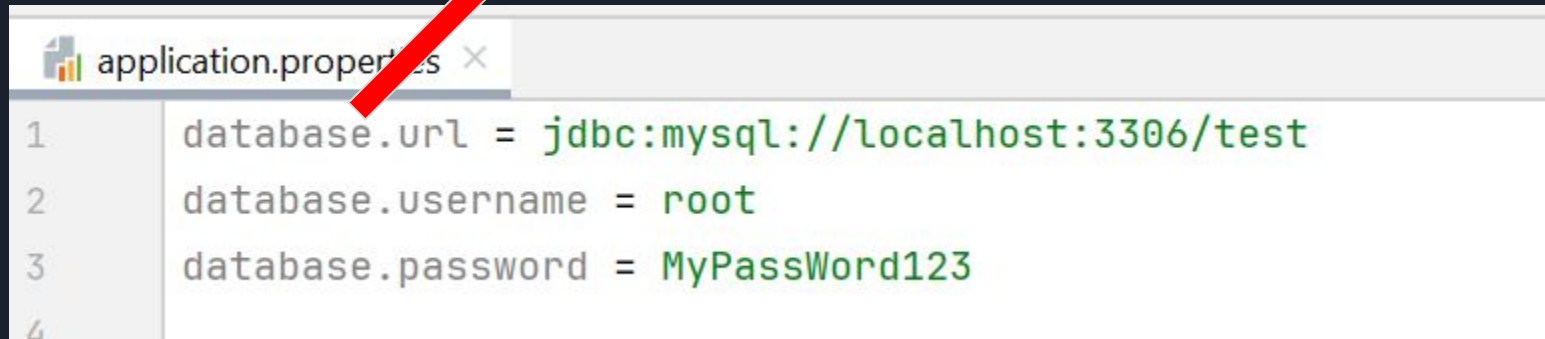
## Step 3: Reference Values from Properties File



The screenshot shows an IDE window titled 'applicationContext.xml'. The XML content is as follows:

```
13 <!-- databaseBean -->
14
15 <bean id="databaseOperationsBean" class="com.dependencyInjection.InjectLiteralValues.config.DatabaseOperationsBean">
16     <property name="url" value="${database.url}"/>
17     <property name="username" value="${database.username}"/>
18     <property name="password" value="${database.password}"/>
19 </bean>
```

A red arrow points from the 'value' attribute of the 'username' property in the XML to the 'application.properties' file shown in the next block.



The screenshot shows an IDE window titled 'application.properties'. The properties are listed as follows:

```
1 database.url = jdbc:mysql://localhost:3306/test
2 database.username = root
3 database.password = MyPassWord123
4
```



**Let's write some code**



**Questions ?**



**THANK YOU**