

@Configuration

```
/*
Spring @Configuration annotation is part of the spring core framework.
Spring Configuration annotation indicates that the class has @Bean definition
methods. So Spring container can process the class and generate Spring Beans
to be used in the application.
*/
no usages
```

ProjectConfig.java
pom.xml (Example1)
Vehicle.java
Example1.java

```

1 package com.example.config;
2
3 import com.example.beans.Vehicle;
4 import org.springframework.context.annotation.Bean;
5 import org.springframework.context.annotation.Configuration;
6
7 projectconfig -- will have details for Spring
8 @Configuration 2 usages
9 public class ProjectConfig {
10
11     @Bean no usages
12     Vehicle vehicle()
13     {
14         var veh=new Vehicle();
15         veh.setName("Audi");
16         return veh;
17     }
18
19     @Bean no usages
20     String hello()
21     {
22         return "Hello World";
23     }
24
25     @Bean no usages
26     Integer number()
27     {
28         return 10;
29     }
30 }
```

ProjectConfig.java
pom.xml (Example1)
Vehicle.java
Example1.java

```

1 package com.example.main;
2
3 import com.example.beans.Vehicle;
4 import com.example.config.ProjectConfig;
5 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
6
7 public class Example1 {
8
9     public static void main(String[] args) {
10         Vehicle vehicle=new Vehicle();
11         vehicle.setName("Honda");
12         System.out.println(vehicle.getName());
13
14         var context=new AnnotationConfigApplicationContext(ProjectConfig.class);
15         Vehicle veh=context.getBean(Vehicle.class);
16         System.out.println(veh.getName());
17
18         /*
19         We don't need to do any explicit casting while fetching a bean from context
20         Spring is smart enough to look for a bean of the type you requested in its
21         If such a bean doesn't exist,Spring will throw an exception.
22         */
23         String val=context.getBean(String.class);
24         System.out.println("Spring Context Val "+val);
25
26         Integer intVal=context.getBean(Integer.class);
27         System.out.println("Spring Context intVal "+intVal);
28     }
29 }
```

Configuration files will have details which class has to considered as Bean and few other details.

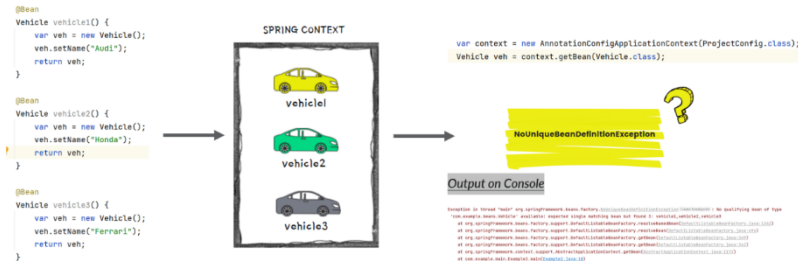
The name of the method will be considered as bean name.

16.Understanding NoUniqueBeanDefinition in Spring

NoUniqueBeanDefinitionException

easy
bytes

When we create multiple objects of same type and try to fetch the bean from context by type, then Spring cannot guess which instance you've declared you refer to. This will lead to `NoUniqueBeanDefinitionException` like shown below,



To resolve this Exception please see the screenshot below

It is like for same class we have defined three Objects all are present in the context and when trying to get them we are getting using class name. As there are multiple Objects with the same class name Spring will have ambiguity which Object/bean to invoke which will lead to `NoUniqueBeanDefinition` Exception

No qualifying bean of type 'com.example.beans.Vehicle' available: expected single matching bean but found 3: vehicle1,vehicle2,vehicle3

I have created three beans of same data type, which is vehicle with the help of three different methods and inside these three methods I have set different different names for my vehicle like Audi, Honda and Ferrari.

With these being declaration, my spring context will load all those beans during startup and you can see all these beans with the name vehicle one, vehicle two and vehicle three is being maintained by the spring IOC container. Like we are discussing previously, whatever method name that you have mentioned, the same name will be considered as a name of your spring bean.

Now when we try to get the details of the bean using get bean method, you are passing the data type of the bean, which is Vehicle(Name of the Class).

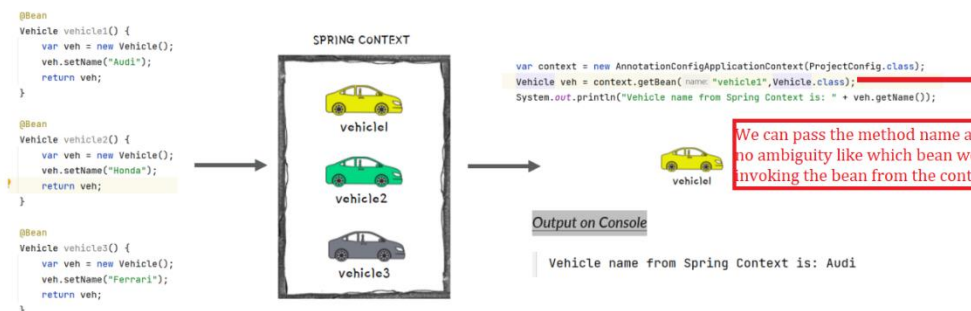
Now at this point of time, my spring has an ambiguity or confusion because there are three beans available inside the spring context with the same data type(With Same ClassName) and the developer is not providing any details other than data type of the bean.

In these kind of scenarios, my spring IOC container will have a confusion and with that confusion it is going to throw an exception saying that **NoUniqueBeanDefinition** exception and you will get an exception

NoUniqueBeanDefinitionException



To avoid `NoUniqueBeanDefinitionException` in these kind of scenarios, we can fetch the bean from the context by mentioning it's name like shown below,



We can pass the method name as well .So that there will be no ambiguity like which bean we were referring to when invoking the bean from the context.

```
Vehicle name from Spring Context is: Audi
```

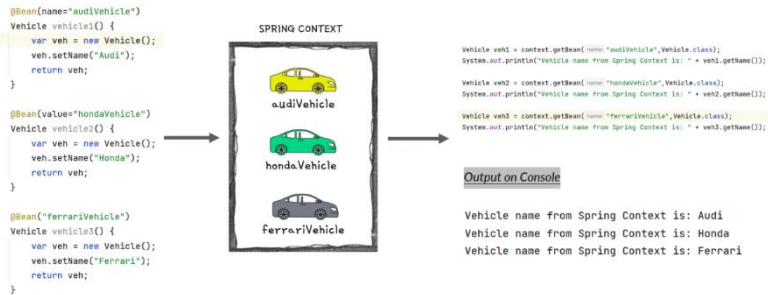
By default whatever is the Java Method that name will be used as bean name.

17. Providing Custom Name to Bean :

DIFFERENT WAYS TO NAME A BEAN

easy
bytes

By default, Spring will consider the method name as the bean name. But if we have a custom requirement to define a separate bean name, then we can use any of the below approach with the help of @Bean annotation.



As we are having issue when we are trying to fetch with the class Type as for multiple class type there were too many multiple returns. We can go by passing the name in the getBean method or

else can Name the bean in the below way.

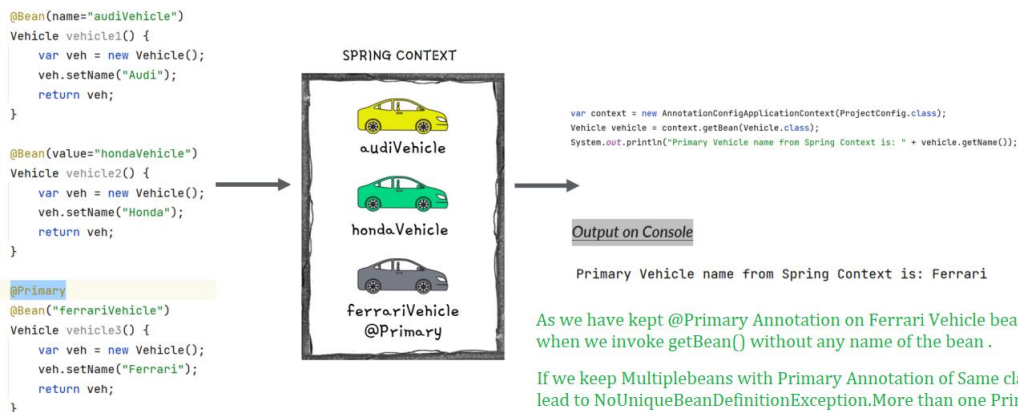
@Bean(name="Name of the Bean1" ----This name will be used as name for that Bean.
@Bean(value="Name of bean")
@Bean("Name of Bean")

18.

@Primary Annotation

easy
bytes

When you have multiple beans of the same kind inside the Spring context, you can make one of them primary by using @Primary annotation. Primary bean is the one which Spring will choose if it has multiple options and you don't specify a name. In other words, it is the default bean that Spring Context will consider in case of confusion due to multiple beans present of same type.



As we have kept @Primary Annotation on Ferrari Vehicle bean it will be considered when we invoke getBean() without any name of the bean .

If we keep Multiplebeans with Primary Annotation of Same class that also will again lead to NoUniqueBeanDefinitionException. More than one Primary Bean found

Primary Annotated Bean will be the default bean that Spring context will consider in case of confusion due to multiple beans of the same type.

19. Creating Beans using @Component Annotation:

@Component Annotation

easy bytes

@Component is one of the most commonly used stereotype annotation by developers. Using this we can easily create and add a bean to the Spring context by writing less code compared to the @Bean option. With stereotype annotations, we need to add the annotation above the class for which we need to have an instance in the Spring context.

Using @ComponentScan annotation over the configuration class, instruct Spring on where to find the classes you marked with stereotype annotations.

```

@Component
public class Vehicle {
    private String name;

    public String getName() {return name;}

    public void setName(String name) {this.name = name;}

    public void printVehicle() {
        System.out.println(
            "Printing Hello from Component Vehicle Bean");
    }
}

```

```

var context = new AnnotationConfigApplicationContext(ProjectConfig.class);
Vehicle vehicle = context.getBean(Vehicle.class);
System.out.println("Component Vehicle name from Spring Context is: " + vehicle.getName());
vehicle.printVehicle();

```

Output on Console

```

Component Vehicle name from Spring Context is: null
Printing Hello from Component Vehicle Bean

```

The name of the bean will be null

we need to add configuration change on top of your project config class since your spring IOC container can't go and scan all of your packages and thousands and thousands of classes available inside your web application to make spring IOC container performance better. So Spring team decided that it's a responsibility of the developer to tell to the spring IOC container a spring framework where it has to go and scan. So how to mention that with the help of other rate component scan? So with the help of @component scan which we are mentioning on top of the configuration class, we are instructing spring, go ahead and look for the Java classes which are annotated with the stereotype annotations like @component. and based upon that, please go ahead and create the spring beans.

20. Stereo Type Annotations in Spring:

Spring Stereotype Annotations

easy bytes

- Spring provides special annotations called Stereotype annotations which will help to create the Spring beans automatically in the application context.
- The stereotype annotations in spring are @Component, @Service, @Repository and @Controller

@Component is used as general on top of any Java class. It is the base for other annotations.

@Service can be used on top of the classes inside the service layer especially where we write business logic and make external API calls.

@Repository can be used on top of the classes which handles the code related to Database access related operations like Insert, Update, Delete etc.

@Controller can be used on top of the classes inside the Controller layer of MVC applications.

Any DAO Classes

So there is no mandate that you need to have @Service, @Repository, @Controller. It's just that it will improve the reliability of your application and all these special annotations.

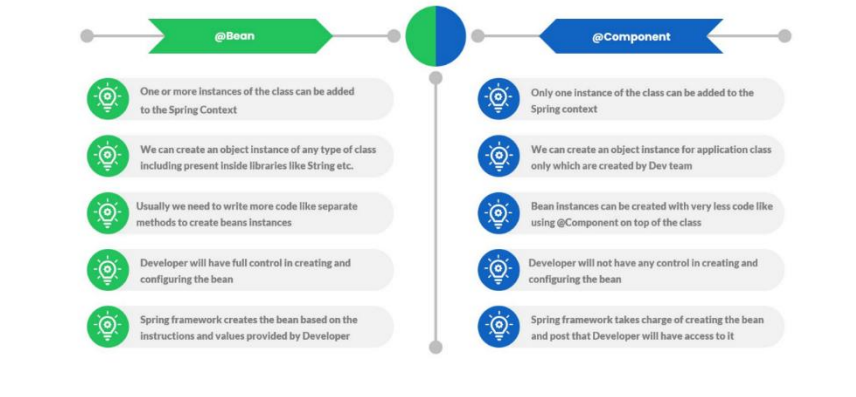
@Service, @Repository and @Controller are extended from the @Component annotation itself.

21. @Bean vs @Component

@Bean Vs @Component

eazy bytes

with the help of @component, we can only create bean object instances for the application classes that are defined by yourself only.



22. @PostConstruct Annotation:

@PostConstruct Annotation

eazy bytes

- We have seen that when we are using stereotype annotations, we don't have control while creating a bean. But what if we want to execute some instructions post Spring creates the bean. For the same, we can use @PostConstruct annotation.
- We can define a method in the component class and annotate that method with @PostConstruct, which instructs Spring to execute that method after it finishes creating the bean.
- Spring borrows the @PostConstruct annotation from Java EE.

```

@Component
public class Vehicle {
    private String name;

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    @PostConstruct
    public void initialize() {
        this.name = "Honda";
    }

    public void printHello(){...}
}

@Configuration
@ComponentScan(basePackages = "com.example.beans")
public class ProjectConfig {
}
        
```

SPRING CONTEXT

```

var context = new AnnotationConfigApplicationContext(ProjectConfig.class);
Vehicle vehicle = context.getBean(Vehicle.class);
System.out.println("Component Vehicle name from Spring Context is: " + vehicle.getName());
vehicle.printHello();
        
```

Output on Console

Component Vehicle name from Spring Context is: Honda
Printing Hello from Component Vehicle Bean

We need to add the below dependency to the application(pom.xml):

```
<!-- https://mvnrepository.com/artifact/jakarta.annotation/jakarta.annotation-api -->
```

```
<dependency>
```

```
<groupId>jakarta.annotation</groupId>
```

```
<artifactId>jakarta.annotation-api</artifactId>
```

```
<version>2.1.1</version>
```

```
</dependency>
```

23. @PreDestroy Annotation:

@PreDestroy Annotation

eazy
bytes

- @PreDestroy annotation can be used on top of the methods and Spring will make sure to call this method just before clearing and destroying the context.
- This can be used in the scenarios where we want to close any IO resources, Database connections etc.
- Spring borrows the @PreDestroy annotation also from Java EE.

```
@Component
public class Vehicle {

    private String name;

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    @PreDestroy
    public void destroy() {
        System.out.println(
            "Destroying Vehicle Bean");
    }

    @Configuration
    @ComponentScan(basePackages = "com.example.beans")
    public class ProjectConfig {

    }
}
```



```
var context = new AnnotationConfigApplicationContext
    (ProjectConfig.class);
Vehicle vehicle = context.getBean(Vehicle.class);
System.out.println("Component Vehicle name from " +
    "Spring Context is: " + vehicle.getName());
vehicle.printHello();
context.close();
```

Output on Console

```
Component Vehicle name from Spring Context is: Honda
Printing Hello from Component Vehicle Bean
Destroying Vehicle Bean
```

23. Creating Beans Programmatically using registerBean() :

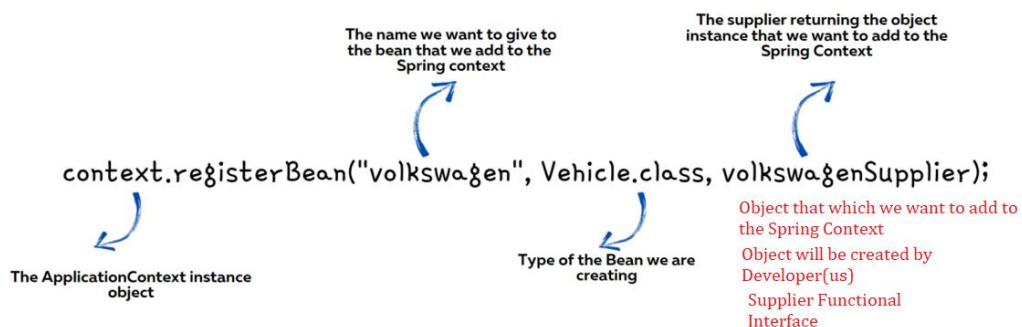
Helps developers in creating the beans based on some programmatic conditions.

Supplier Interface that doesn't take any input but it returns an Object.

ADDING NEW BEANS PROGRAMMATICALLY

eazy
bytes

- Sometimes we want to create new instances of an object and add them into the Spring context based on a programming condition. For the same, from Spring 5 version, a new approach is provided to create the beans programmatically by invoking the `registerBean()` method present inside the context object.



```
package com.example.main;

import com.example.beans.Vehicle;
import com.example.config.ProjectConfig;
import org.springframework.beans.factory.NoSuchBeanDefinitionException;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import java.util.Random;
import java.util.function.Supplier;
```

```

public class Example8 {

    public static void main(String[] args) {

        var context=new AnnotationConfigApplicationContext(ProjectConfig.class);
        Vehicle shine=new Vehicle();
        shine.setName("Shine");
        Supplier<Vehicle> shineSupplier=() ->shine;

        Supplier<Vehicle> javaSupplier=() ->{
            Vehicle javaSup=new Vehicle();
            javaSup.setName("Java_Bike");
            return javaSup;
        };
        Vehicle shineBike= null;
        Vehicle javaBike = null;
        Random rnd=new Random();
        int randomNumber=rnd.nextInt(10);
        if(randomNumber%2==0)
        {
            context.registerBean("Java Bike", Vehicle.class,javaSupplier);
        }
        else
        {
            context.registerBean("Shine Bike", Vehicle.class,shineSupplier);
        }
        Vehicle veh=context.getBean(Vehicle.class);
        System.out.println("Random number "+randomNumber);
        System.out.println("Vehicle from the Spring Container "+veh.getName());

        try {
            shineBike = context.getBean("Java Bike",Vehicle.class);
        }catch (NoSuchBeanDefinitionException noSuchBeanDefinitionException){
            System.out.println("Error while creating Volkswagen vehicle");
        }
        try {
            javaBike = context.getBean("Shine Bike",Vehicle.class);
        }catch (NoSuchBeanDefinitionException noSuchBeanDefinitionException){
            System.out.println("Error while creating Audi vehicle");
        }

        if(null != shineBike){
            System.out.println("Programming Vehicle name from Spring Context is: " +
            shineBike.getName());
        }else{
            System.out.println("Programming Vehicle name from Spring Context is: " +
            javaBike.getName());
        } context.close();
    }
}

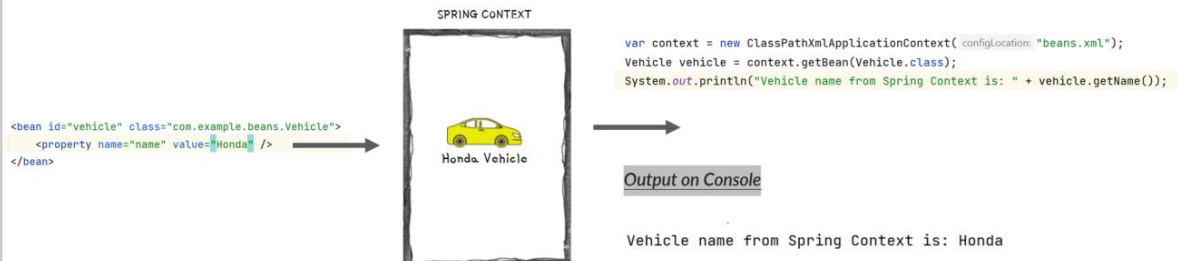
```

25.Creating Beans using XML Configurations:

ADDING NEW BEANS USING XML CONFIGS

easy
bytes

- In the initial versions of Spring, the bean and other configurations used to be done using XML. But over the time, Spring team brings annotation based configurations to make developers life easy. Today we can see XML configurations only in the older applications built based on initial versions of Spring.
- It is good to understand on how to create a bean inside Spring context using XML style configurations. So that, it will be useful if ever there is a scenario where you need to work in a project based on initial versions of Spring.



ADDING NEW BEANS USING XML CONFIGS

easy
bytes

- In the initial versions of Spring, the bean and other configurations used to be done using XML. But over the time, Spring team brings annotation based configurations to make developers life easy. Today we can see XML configurations only in the older applications built based on initial versions of Spring.
 - It is good to understand on how to create a bean inside Spring context using XML style configurations. So that, it will be useful if ever there is a scenario where you need to work in a project based on initial versions of Spring.
- if you have a scenario where you have initial versions of spring like one and two, then definitely there is no annotation support. We have to use XML style of configurations.

whenever you have configurations or some property files, then definitely you need to maintain under resources folder.

configuration file should be kept in the resources folder

Application context is an interface and there are many implementations of it inside spring framework.

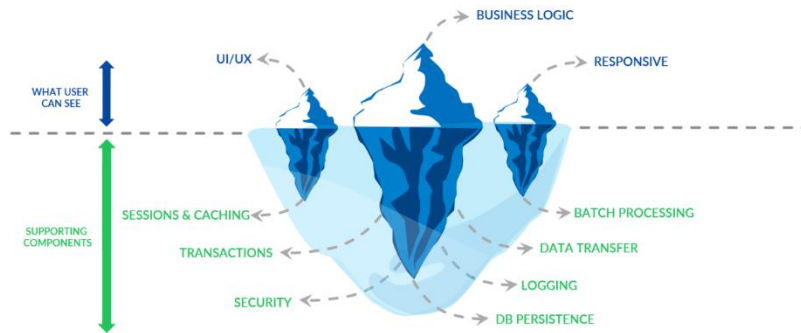
So based upon the scenario that you are into, you can leverage the corresponding implementation of application context.



26. Why should we use frameworks:

BEHIND THE SCENES OF A WEB APP

easy
bytes



As a developer, if I start focusing on building all the supporting components, it will take years and years to build a small web application. So to avoid such scenarios, we have frameworks where they will provide all the supporting components

like you want to implement caching or security or DB persistence logging. The scenario can be anything.

You can leverage these frameworks and they will provide all the common supporting components to the developers.

And it's a duty of the developer to understand that framework and to configure that framework inside your web application.

WHY SHOULD WE USE FRAMEWORKS?

easy
bytes



CHEF VICKY

Uses best readily available best ingredients like Cheese, Pizza Dough etc. to prepare Pizza



Pizza preparation time is less



Can easily scale his restaurant pizza orders



Gets consistent taste for his pizzas



Focus more on the pizza preparation



Less efforts and more results/revenue



CHEF SANJEEV

Prepare all the ingredients like Cheese, Pizza Dough etc. by himself to prepare Pizza



Pizza preparation time is more



Scaling his restaurant pizza orders is not an option



May not get a consistent taste for his pizzas



Focus more on the raw material & ingredients



More efforts and less results/revenue

WHY SHOULD WE USE FRAMEWORKS?

eazy
bytes



DEV SANJEEV

Uses best readily available best frameworks like Spring, Angular etc. to build a web app



Leverage Security, Logging etc. from frameworks



Can easily scale his application



App will work in an predictable manner



Focus more on the business logic



Less efforts and more results/revenue



DEV VICKY

Build his own code by himself to build a web app



Need to build code for Security, Logging etc.



Scaling his is not an option till he test everything



App may not work in an predictable manner



Focus more on the supporting components



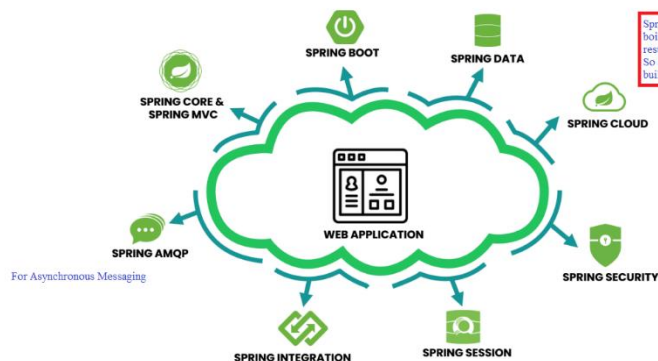
More efforts and less results/revenue

27.Introduction to Spring Projects:

There are many Spring based Projects

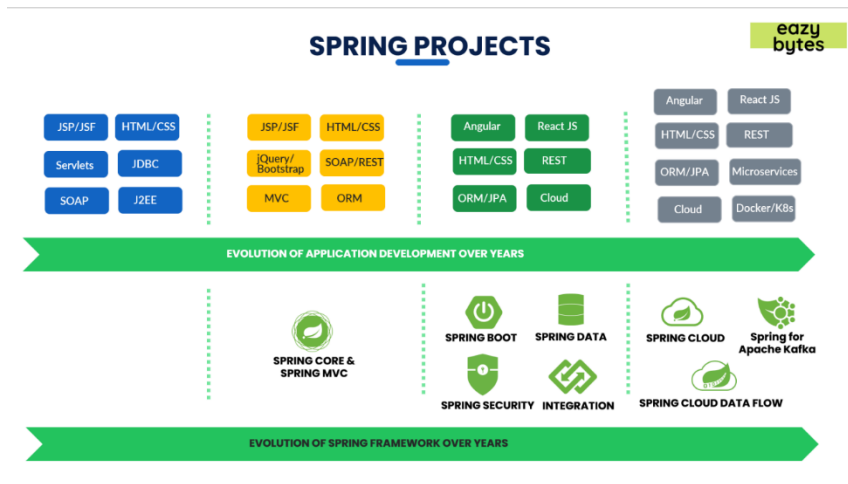
SPRING PROJECTS

eazy
bytes



Spring Data : Developer can write very less code and he can achieve more results by avoiding all the boilerplate code that he needs to write, especially to create a connection, closing a connection, fetching the results, iterating them, handling the transactions. So all those heavy lifting will be done by spring framework, by Spring data project for you. You just focus on building your business logic. That's the purpose of Spring data project.

* These projects are few important projects from Spring but not a complete list.



the official website of spring is spring.io, and spring is being maintained by the VMware.vMware is an organization who maintains and regularly introduces various projects or new features or upgrades to the spring framework.

Spring has many projects and based upon the need that you have and based upon the scenario that you are in, you can choose a specific project and try to adopt that inside your web application.

When we create multiple objects of same class type and try to fetch the bean from context by class type, then Spring cannot guess which instance you've declared you refer to. This will lead to NoUniqueBeanDefinition Exception.

Question 9:

Why should we use frameworks like Spring for web development inside our projects?

- ☐ Because everyone is using and I need to follow the same
- ☐ Frameworks makes my life super easy and I don't need to write any code. Simply deploy the code & monitor it
- ☒ With the help of Frameworks, we can just focus on business logic, since frameworks take care of the common non functional requirements like logging, security etc. Also when frameworks used, it makes our application predictable and easily scalable.