1. **Introduction to Spring Boot:**
2. what problem it solved
3. Its features and advantages

**Setting up initial Spring boot Project Understand Layered Architecture**

1. Understand Layered Architecture

Understand Maven pom.xml a bit (as Maven and Java is a prerequisite for learning springboot)

**Spring Boot Basic Topics**

Annotations:

@SpringBootApplication

@ Controller

@RestController

@RequestMapping

@GetMapping, @PostMapping, @PutMapping, @DeleteMapping

@Autowired

@Component

@Service

@Repository

@ComponentScan

@ Configuration

@Value

@Qualifier

@Profile

@EnableAutoConfiguration

@Entity

@Transactional

@EnableCaching

@Async

@EnableScheduling etc…

# Introduction to Spring Boot:

What is servlet:- Before we talk about Spring or Spring boot, first we need to discuss about “SERVLET” and “Servlet Container”

* Provides foundation for building web applications.
* Servlet is a class, which handles client request, process it and return the response.
* And servlet container are the ones which manages the Servlets.

A diagram of a server

Description automatically generated

Servlet container

## what problem it solved

Spring Framework, solve challenges which exists with Servlets.

1. Removal of web.xml
   1. This web.xml over the time becomes too big and becomes very difficult to manager and understand.
   2. Spring framework introduced Annotations based configuration.
2. Inversion of Control(IOC):-
   1. Servlets depends on Servlet container to create object and maintain its lifecycle.
   2. IoC is more flexible way to manage object dependencies and its lifecycle (through Dependency injection)
   3. Dependency Injection is implementation of Ioc
3. Unit testing is much harder:
   1. As the object creation depends on Servlet, mocking is not easy. Which makes unit testing process harder.
   2. Spring dependency injection facility makes the Unit testing very easy.
4. Difficult to manage REST APIs
   1. Handling different HTTP methods, request parameters, path mapping make code little difficult to understand.
   2. Spring MVC provides an organized approach to handle the requests and its easy to build RESTful APIs.

There are many other areas where spring framework makes developer life such as: integration with other technology like hibernate, adding security, etc.

1. The another important feature of Spring framework is lot of INTEGRATION available with other frameworks.
   1. This allow Developers to choose different combination of technologies and frameworks which best fits their requirements like:
      1. Integration with Unit Testing framework like Junit or Mockito.
      2. Integration with Data Access framework like Hibernate, JDBC, JPA etc.
      3. Integration with Asynchronous programming.
      4. Similar way, it has different integration available for:
         1. Caching
         2. Messaging
         3. Security etc.

**IOC Inversioin of Control:-**

**@component:-** Tells Spring that, you have to manage this class or bean.

@Autowired:- Tells Spring to resolve and add this object dependency.

**Spring Architecture:-**

A diagram of a system

Description automatically generated

Fig:- Spring MVC Architecture

@controller

@RequestMapping("/Paymentsapi")

public class PaymentController{

    @Autowired

    PaymentDAO paymentService;

    @GetMapping("/payment")

    public String getPaymentDetails(){

        return paymentService.getDetails();

    }

}

To write a basic Spring application we must have

1. Controller
2. Dispatcher class
3. Pom.xml
4. Config class

## Advantages of Spring Boot over Spring

Spring Boot, solve challenges which exists with Spring MVC.

1. Dependency Management:- No need for adding different dependencies separately and also their compatible version headache.
2. Auto Configuration:- No need for separately configuring “DispatcherServelt”, “AppConfig”, “EnableWebMVC”, “ComponentScan”. Spring boot add internally by default.
3. Embedded Server:- In traditional Spring MVC application, we need to build a WAR file, which is a packaged file containing your application’s classes, JSP pages, configuration files, and dependencies. Then we need to deploy this WAR file to a servlet container like Tomcat.

But in Spring boot, Servlet container is already embedded, we don’t have to do all this stuff. Just run the application, that’s all.

## So, What is Spring boot?

1. It provides a quick way to create a production ready application.
2. It is based on Spring framework.
3. It support “Convention over Configuration”.
   1. Use default values for configuration, and if developer don’t want to go with convention( the way something is done), they can override it.
4. It also help to run an application as quick as possible.

# Project Setup and Layered Architecture

1. Go to Spring Initializer i.e “start.spring.io”
2. Select Language as Java, Maven
3. Spring Boot : 3.2.3
4. Project Metadata
   1. Group - Com.PrashanthVangari
   2. Artifact – learningspringboot - // Usually project name
5. War vs Jar ( Web Archive vs Java Archive)
   1. Where we need stand alone application we use JAR.
   2. If we use HTML, CSS, Javascript then we need to use.
   3. We use Jar these days.
6. Java 17
7. Dependencies:-
   1. Spring web

A screenshot of a computer

Description automatically generated

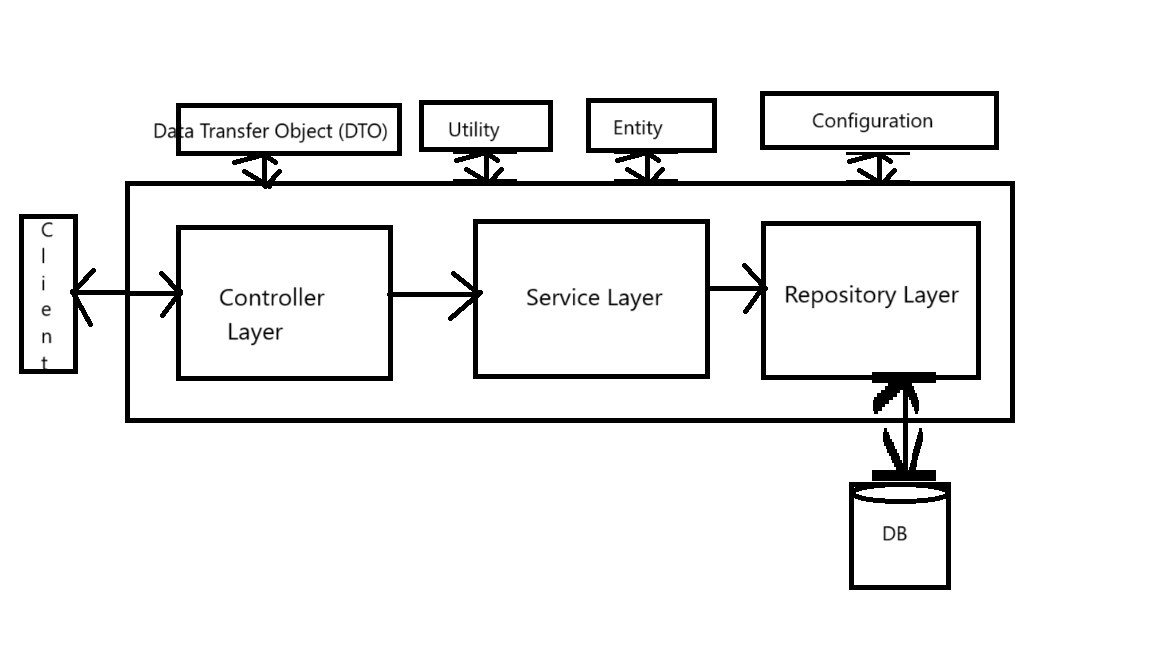


Fig Layered Architecture

1. Service Layer has
2. Entity is the class which is the direct representation of our tables in the sql. Generally entities are used by repository layer.

# Introduction to Maven

1. It is a project management tool. Helps developers with:
   1. Build generation
   2. Dependency resolution
   3. Documentation etc
2. Maven uses POM (Project Object Model) to achieve this.
3. When “maven” command is given, it looks for “pom.xml” in the current directory and get needed configuration.

A computer screen shot of a black background

Description automatically generated

Fig Maven Project Structure

## How to read pom.xml file:

1. Each pom.xml file has parent pom. If <parent> is not present in the xml file then it inherits from the super pom
2. groupId, artifactId, version these uniquely identifies the project.

### Build LifeCycle:

1. If you want to run “package” phase, all its previous phase will get executed first.
2. And if you want to run specific goal of a particular phase, then all the goals of previous phase + current phase goals before the one you defined will get run.

A diagram of a process

Description automatically generated with medium confidence

Fig: Phases in the maven

1. Validate project structure
2. Compile source code
3. Test the code (unit test)
4. Package Compiled code (ex:- Jar or War)
5. Verify the integrity of package
6. Install the package in local repository
7. Deploy the package in Remote Repository
8. So, each phase has set of goals to achieve before moving to the next phase. If we want to add any task/goal to any of the phase we use build
   1. Validate project structure: Maven already has Validate phase defined and its goal, but if we want to add more goals or tasks, then we can use<build> element. And add the goal to specific phase.
      1. Validate: ***mvn validate***
   2. Compile:-
      1. ***mvn compile***
      2. compiles the code and creates a target folder which has all the class files.
   3. Test:-
      1. ***mvn test***
      2. This includes validate, compile and test phases.
   4. Package:-
      1. ***mvn package***
      2. First compile validate, compile, test and then run package phase in which it generates .jar or .war file.
      3. Jar or war is present in the target folder.
   5. Verify the integrity of the package:-
      1. Does not force to verify (optional)
      2. It can perform some additional checks apart from unit test cases like:
         1. Static code analysis: PMD is a source code analyzer.
            1. Finds unused variable
            2. Finds unused imports
            3. Empty catch block
            4. No usage of object
            5. Finds duplicate code etc.
         2. Check sum verification etc.
   6. Install
      1. ***mvn install***
      2. It will install the .jar file in the local Maven repository.
      3. Which is typically located in your user home directory(~/.m2/repository)
   7. Deploy
      1. mvn deploy
      2. it will deploy the .jar to REMOTE Repository
      3. In the settings.xml file we have to give the repository credentials.
      4. We can also deploy it to the maven central repository.

A screenshot of a computer program

Description automatically generated

# Spring boot Annotations

## Java Annotations:

## @Controller:

1. It indicates that the class is responsible for handling incoming HTTP requests.

A screen shot of a computer program

Description automatically generated

1. Spring boot will include this class in the list of controller classes. Then the spring boot will decide if they work on the http request.

## @ResponseBody

1. In the above image ResponseBody is responsible to let the springboot know that the returning value is a responsebody and not the view.
2. Generally, the springboot searches for the view with that name, in our case “fetching and returning user details” and then it returns the view. But by mentioning responsebody we will tell the springboot that the returning is a response body.

## @RestController:

1. RestController does not have the @ResponseBody because @RestController is @Controller + @ResponseBody

A screenshot of a computer program

Description automatically generated

## @RequestMapping:

1. It maps the request to the method.
2. In the image

A screenshot of a computer program

Description automatically generated

## @GetMapping

1. This is used instead of @RequestMapping. We need not specify the ‘method= RequestMethod.GET’ as it is already present in the @GetMapping

A screenshot of a computer program

Description automatically generated

## @RequestParam:

1. Used to bind, request parameter to controller method parameter.
2. We have required parameter as true by default.

A computer screen shot of a computer code

Description automatically generated

1. The framework automatically peforms type conversion from the request parameter’s string representation to the specified type.
   1. Primitive types: such as int, long, float, double, Boolean, etc.
   2. Wrapper Classes: Such as Integer, Long, Float, Double, Boolean, etc.
   3. String: Request parameters are inherently treated as strings only.
   4. Enums: You can bind request parameters to enum types.
   5. Custom object types: We can do it using a registered PropertyEditor.

## @InitBinder

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

## @PathVariable

1. Used to extract values from the path of the URL and help to bind it to controller method parameter.

A screen shot of a computer code

Description automatically generated

## @RequestBody

1. Bind the body of HTTP request (typically JSON) to controller method parameter( java object).

A screenshot of a computer program

Description automatically generated

1. GSON or Jackson library is used to convert the request body details to a java object.

A screenshot of a computer program

Description automatically generated

## @ResponseEntity

1. It represents the entire HTTP response.
2. Header, Status, response body etc.

A screenshot of a computer

Description automatically generated

# Spring Beans

## What is a Spring Beans

1. Bean = Object

Objects without spring beans.

1. Single instance
2. Multiple instance
3. Repository Behavior
4. Controller Behavior
5. Service Behavior
6. ….

Benefits of having behavioral objects.

1. More structural
2. Avoids code littering and makes it easily understandable.
3. Easy to debug.
4. Easy to maintain the application.

Developer can implement them but not needed and the reasons are ?:

1. More complicated and takes time.
2. Not the actual business logic
3. Companies focus more on the business logic.
4. Need not reinvent the wheel when it is already available.

## Spring beans creation

1. There are three ways to create beans
   1. Java configuration
   2. Annotation
   3. Xml configuration
2. More information on each way:
   1. Java Configuration:

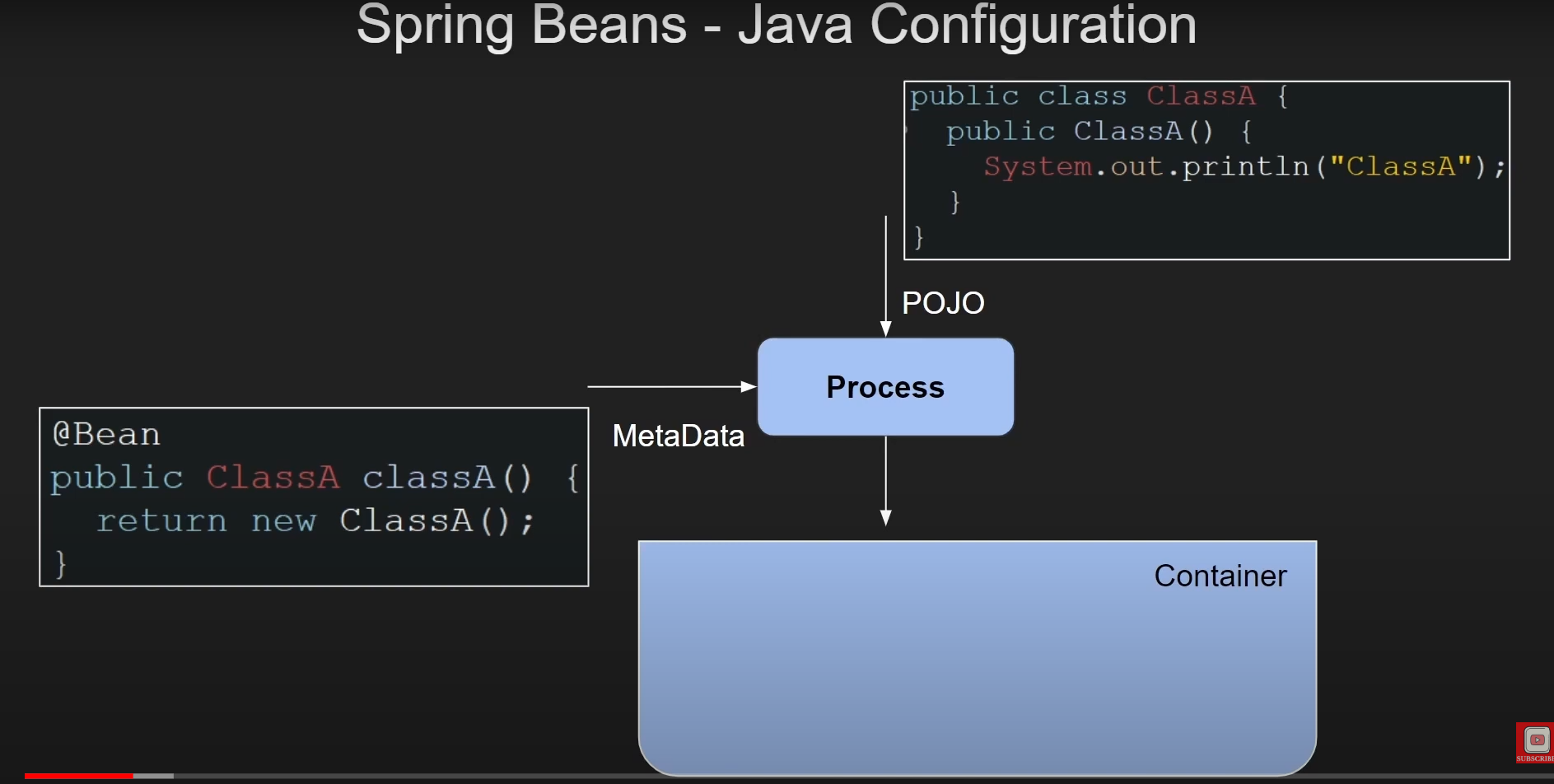
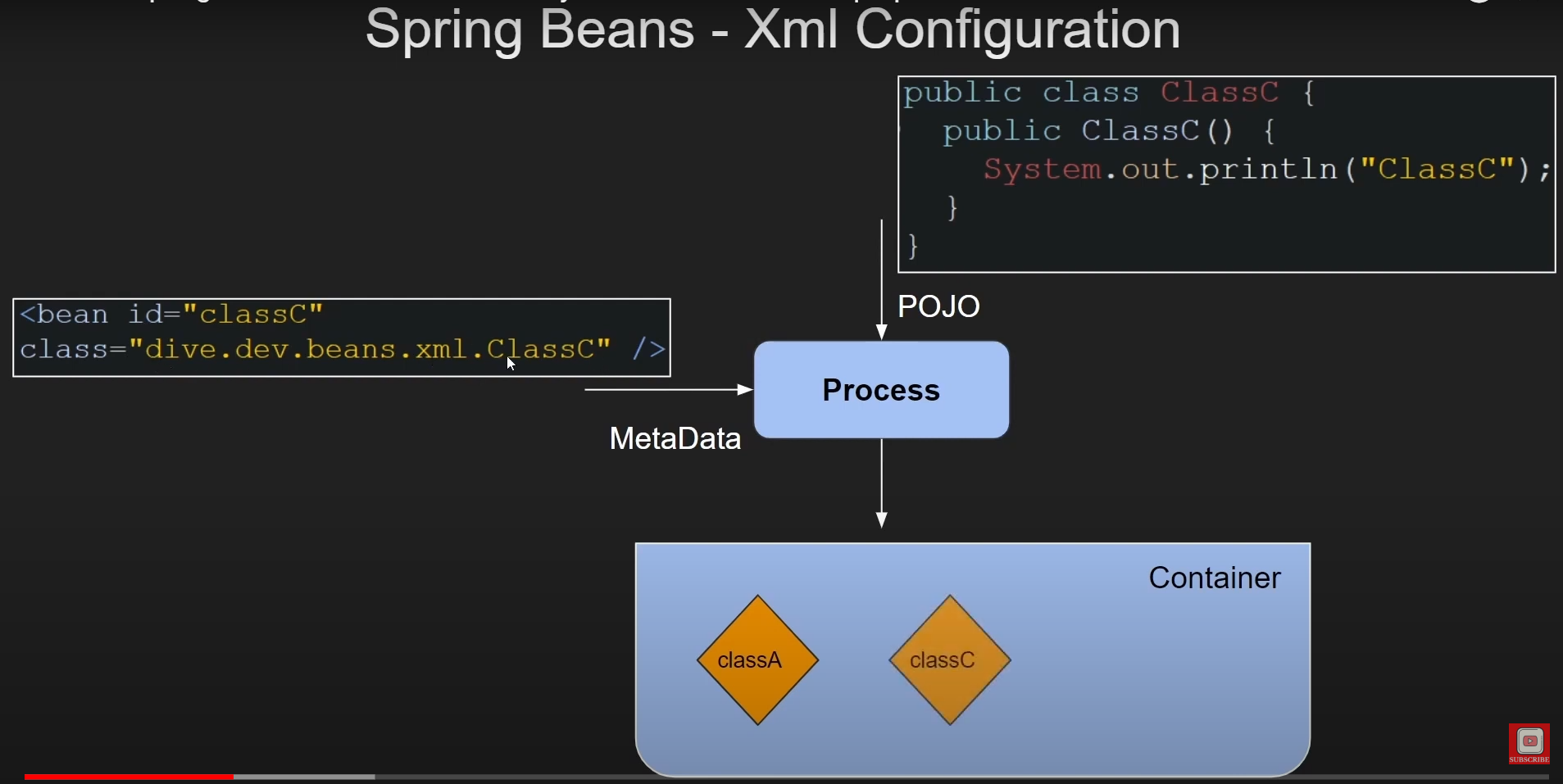


Fig: Java Configuration

* 1. Xml configuration:



* 1. Annotation:

A computer screen shot of a diagram

Description automatically generated

## Life Cycle of Beans

|  |  |
| --- | --- |
| Life cycle consists of mainly 4 stages:   1. Bean Creation – Scan the application to create beans. Create beans and there dependencies. 2. Invoke init() – call the initialization method(if configured)    1. Defined in three different ways for different configurations. Invoked once the bean and its dependencies are created. 3. Use beans in Application – Using the beans in the application according to the requirements. 4. Invoke destroy() – Call destroy method (if configured) Defined in three different ways for different configurations. Invoked just before the application shutdown. | A diagram of a process  Description automatically generated |

1. Beans creation:
   1. Beans configuration
      1. Scan the application to register beans
      2. Bean name in list<beanName>
      3. Bean information in registerbean<beanName, BeanDefinition>
      4. Bean alias information in registerAliasses<alias, beanName>
   2. Beans instantiation
      1. Gets bean name from the list<beanName>
      2. Create instance of bean.
      3. Initialize the dependent beans.
      4. Invoke the initialization methods.
   3. Components Scan:
      1. Scan the application and find classes annotated with @Component, @ManagedBean, @Named

A screenshot of a computer program

Description automatically generated

1. Core features of the Spring:
   1. IOC – inversion on control
   2. AOP – Aspect oriented Programming
   3. DAF – Data access Framework