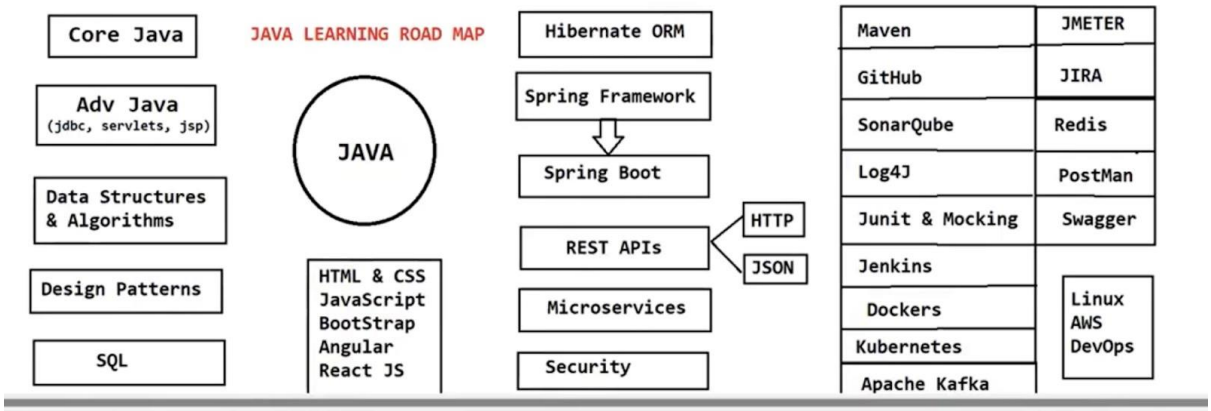


++++++++++++++++++++++++

Application Development

++++++++++++++++++++++++

=> Application development divided into 2 parts

1. Backend Development
2. Frontend Development

Fullstack Development => Backend development + Frontend development

=> Backend contains the business logic of our application
(Webservice calls, validations, email logic, b communication).

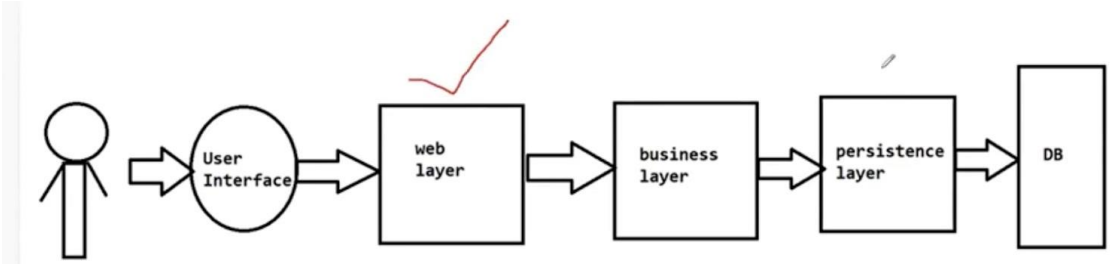
=> Frontend contains user interface of the application
(Presentation logic).

Spring Framework

1. What is Programming Language
2. What is Framework

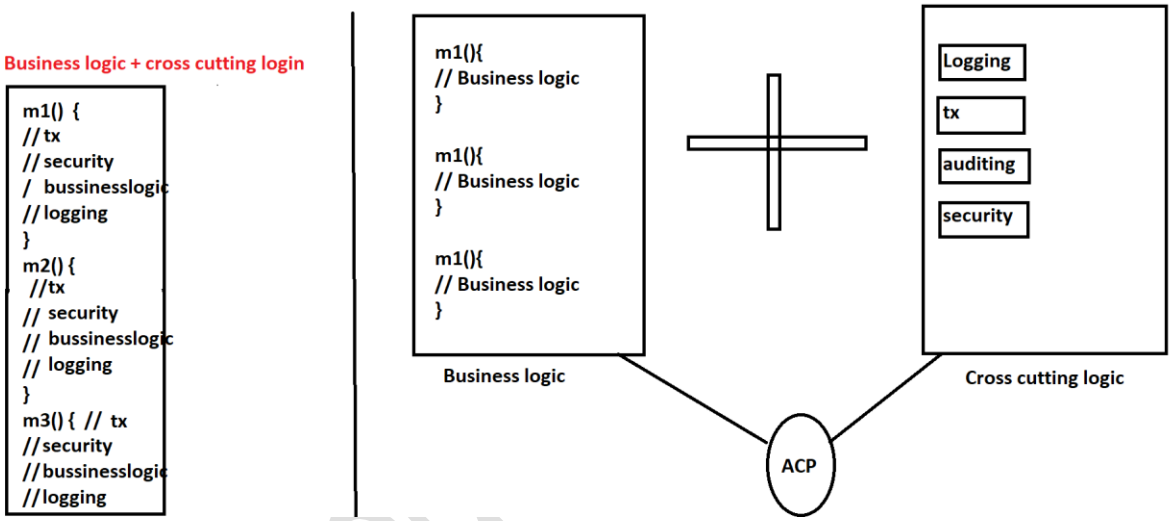
- Programming Language used by humans to communicate with Computers.
- Programming Language contains set of instructions
  Ex: C, C++, Java, C#, Python etc.....
- Framework means semi-developed software.
- Frameworks provides some common logics which layer required for application development

- Ex: Hibernate, Struts, Spring etc...
- If we use framework in our application development then we need focus only on business logic (framework will provide common logics).



Hibernate: It is an ORM framework. Used to develop persistence layer of our application.
Struts: It is a web framework. Used to develop web layer of our application.

Dimag-3



Spring: It is an application development framework. Entire project can be developed by using this.

Spring Advantages

- It is a free & open source framework
- Spring is very light weight framework
- Spring is versatile framework

(Spring can be integrated with any other java framework available in the market).

Spring is non-invasive framework
(Spring framework will not force us to use framework related interfaces or classes)
Ex: To create a servlet we need to implement Servlet Interface or we need to HttpServlet or GenericServlet. In Spring we can create a simple pojo then spring will execute our pojo classes.

Note: In Spring we can create a simple pojo and we can ask spring to execute our pojo

Spring works based on POJO and POJI model
POJO: Plain old java object
POJI: Plain old java interface

Spring is not a single framework. It is collection of Modules
Spring Modules

1. Spring Core
2. Spring Context
3. Spring DAO / Spring JDBC
4. Spring AOP
5. Spring ORM
6. Spring Web MVC
7. Spring Security
8. Spring REST
9. Spring Data
   Spring Cloud
   Spring Batch etc...

Note: Spring framework is loosely coupled. It will not force to use all modules.
-> Based on project requirement we can choose which modules we need to use from spring.

Note: Spring Core is the base module for all other modules in spring. To work with any module in spring first we need to know Spring Core module.

-> Spring Core Module is base module in Spring Framework. It is providing IOC container & Dependency Injection.

Note: IOC & DI are fundamental concepts of Spring Framework.

- Spring Context module will deal the configuration related stuff

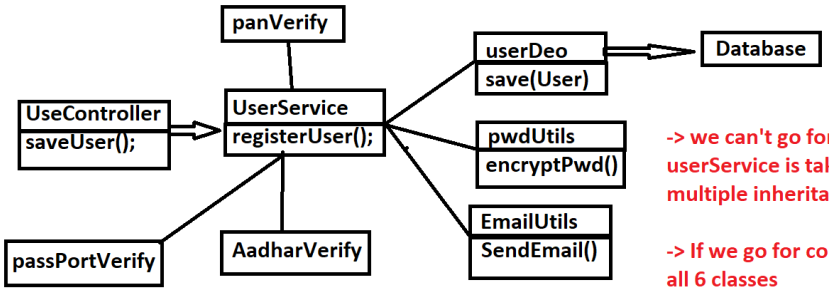
- AOP stands for Aspect Oriented Programming. Spring AOP is used to deal with Cross Cutting logics in application.

- Application = Business Logic + Cross Cutting Logic

Note: We can seperate business logic and cross cutting logic using AOP module.

Dimag-4

Requirment



-> we can't go for inheritance here because userService is taking to 6 classes (java does't support multiple inheritance).

-> If we go for composition we should obj creation for all 6 classes

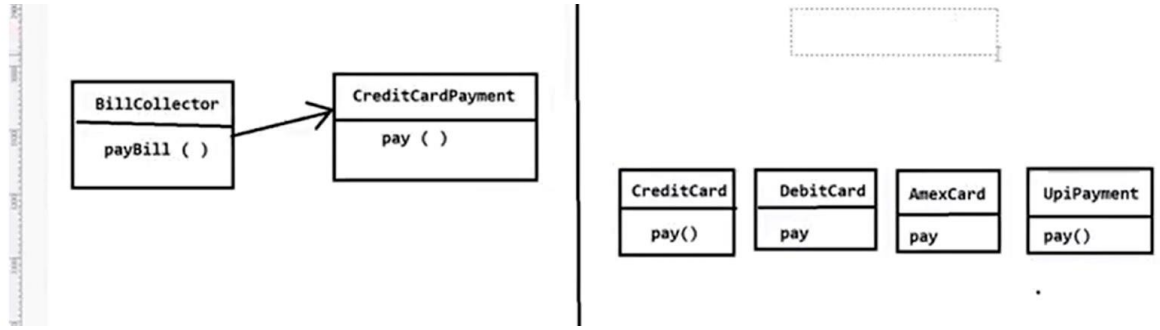
Note: in both approaches classes will become tightly coupled.

- Spring JDBC / Spring DAO module used to develop Persistence Layer
- Spring ORM module is used to develop Persistence Layer with ORM features.
- Spring Web MVC Module is used to develop Web Applications.
- Spring Security module is used to implement Security Features in our application (Authentication & Authorization).
- Spring REST is used to develop RESTFul services (REST API)
- Spring Data is used to develop persistence layer. It provided pre-defined repositories to simplify CRUD operations.
- Spring Cloud providing Cloud concepts like Service Registry, Cloud Gateway, Filters, Routing, FeignClient etc..
- Spring Batch is used to implement Batch Processing in our application. Batch Processing means bulk operation.
- Spring Framework released in 2004 (First Production Released)
- The current version of Spring is S. version (Released in 2017)

Note: Reactive Programming support added in Spring Framwork 5. version
Note: Spring Boot 1. released in 2014

-> The current version of Spring Boot is 3. version
Note: Spring Boot is an extension for Spring Framework.

Diag-5



Dimag-6-

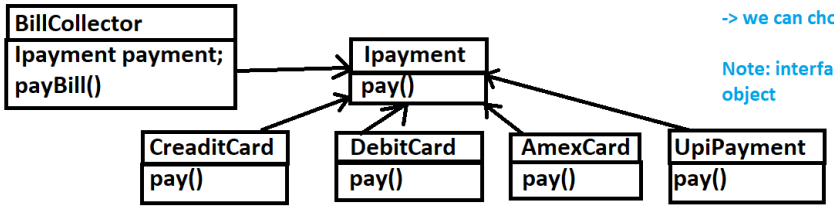
Taking to interface instead of implementation classes

->BillCollector having interface reference variable

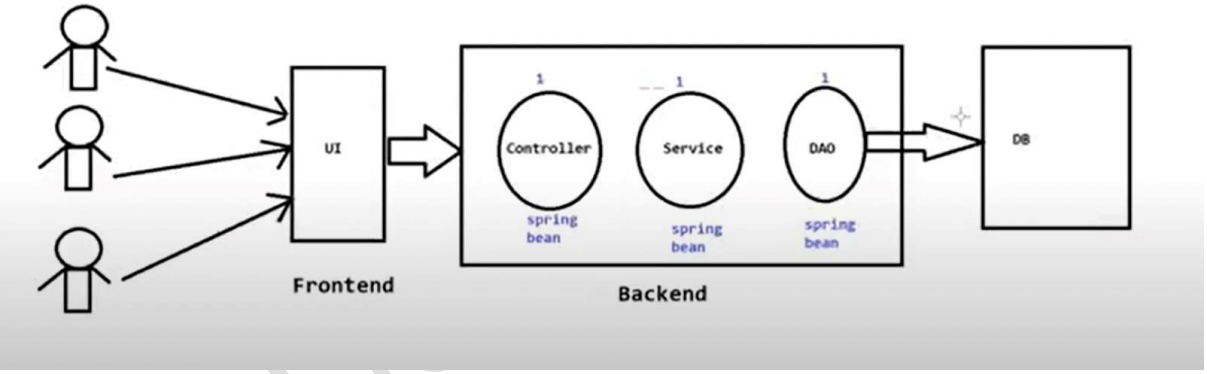
->Interface having multiple implementation

-> we can choose any implementation in runtime

Note: interface ref variable can hold impl class object



dimag-7



File Edit

- If we want Spring Core Module to manage dependencies among the classes with loosely coupling then we have to develop our classes by following some best practises.

- "Spring Core" suggesting Developers to follow "Strategy Design Pattern" to develop classes so that

"Spring Core" can easily manage dependencies among the classes with loosely coupling.

Strategy Design Pattern

- It comes under Behavioural Design Pattern
- It enables our code to choose an alogrithm at run time

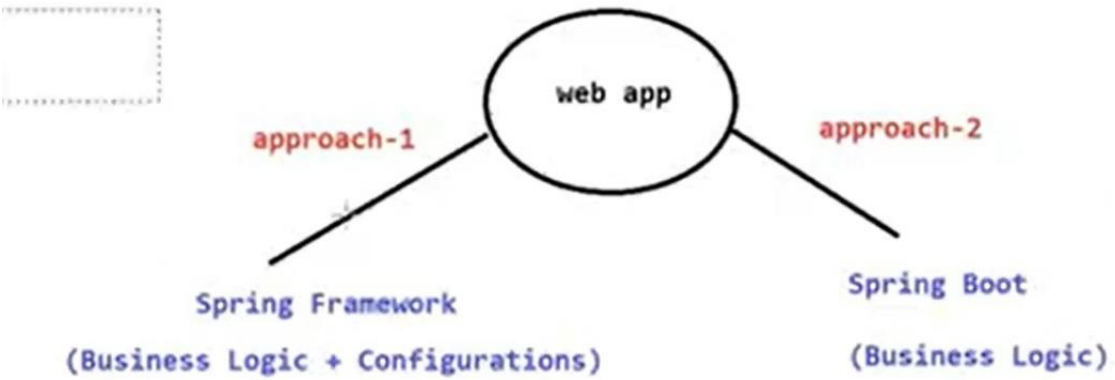
Strategy Design Pattern

- It comes under Behavioural Design Pattern
- It enables our code to choose an alogrithm at run time
- When we have multiple algorithms then we can use this pattern to choose one algorithm at run time

Rules

1.
2. Favour Compositon over inhertience
3. Code to interfaces instead of implementation classes
4. Code should be open for extension and code should be closed for modification

Diag-8:



Code pending

Dependency Injection

==========

-> The process of injecting one class object into another class object is called as Dependency Injection

-> We can perform Dependency Injection in 3 ways

1. Setter Infection
2. Constructor Injection
3. Field Injection

-> The process of injecting one class object into another class object using Setter method then it is called as Setter Injection.
-> We can perform Dependency Injection in 3 ways

1. Setter Injection
2. Constructor Injection
3. Field Injection

-> The process of injecting one class object into another class object using Setter method then it is called as Setter Injection.
Ex::

BillCollector bc = new BillCollector);
bc. setPayment (new CreditCardPayment ());

-› The process of injecting one class object into another class object using Constructor is called as Constructor Injection

-> The process of injecting one class object into another class object using Constructor is called as Constructor Injection

Ex::
BillCollector bc1 = new BillCollector (new DebitcardPayment));
-> The process of injecting one class object into another class objet using variable is called as Field Injection.

Code……pending

- If we develop the project without using spring framework then programmer should perform dependency injection.

- If programmer performs dependency injection then classes will become tightly coupled.

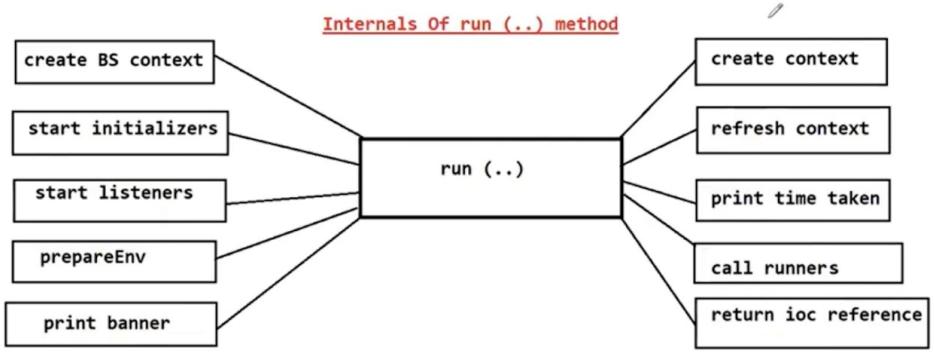
- If we develop the project using spring framework then Spring IOC will take care of Dependency Injections in our application.

- IOC is a principle which is responsible to manage and colloborate dependencies among the classes available in the application.

Note: IOC stands for Inversion of Control. DI stands for Dependency Injection.

-> In spring framework IOC will perform DI.

Dimag-8



Building First Application Using Spring Framework

1. Open STS IDE and Create Maven Project
2. Add Spring-Context dependency in project pom.ml file

‹dependencies>

‹dependency>
‹groupId>org.springframework‹/groupId> ‹artifactId>spring-context</artifactId>
‹version›S.3.20</version>
</dependency>

</dependencies>

3) Create required classes in our application Üsing Strategy Design Pattern

IPayment.java
CreditCardPayment. java
DebitCardPayment.java
UpiPayment.java
BillCollector.java
Test.java

public class Test {
public static void main(String[] args) throws Excepti]n {
ApplicationContext context =
new ClassPathXmlApplicationContext("Spring-Beans.xml");
}
}
-> To perform setter injection we will use ‹property /> tag like below
<bean id-"billcollector" class-"in.bytecode.Billcollector">

‹property name "payment" ref_"upi"
< bean>

-> To perform constructor injection we will use <constructor-arg/> tag like blew

<bean id-"billcollector" class-"combytecode.Billcollector"> «constructor-arg name-"payment"
ref-"upi"
< bean>

when we perform both setter and constructor injection for same variable then setter injection will override constructor injection because construcor will execute

‹bean id-"billCollector" class-"in.bytecode.Billcollector"> ‹property name- "payment"
ref."creditcard"
/>
«constructor-arg names "payment" refs"upi" Is
</bean>

Bean Scopes

- > Bean scope will decide how many objects should be create for a spring bean
- > The default scope is singleton (means only one object will be created)
- > We can configure below scopes for spring bean

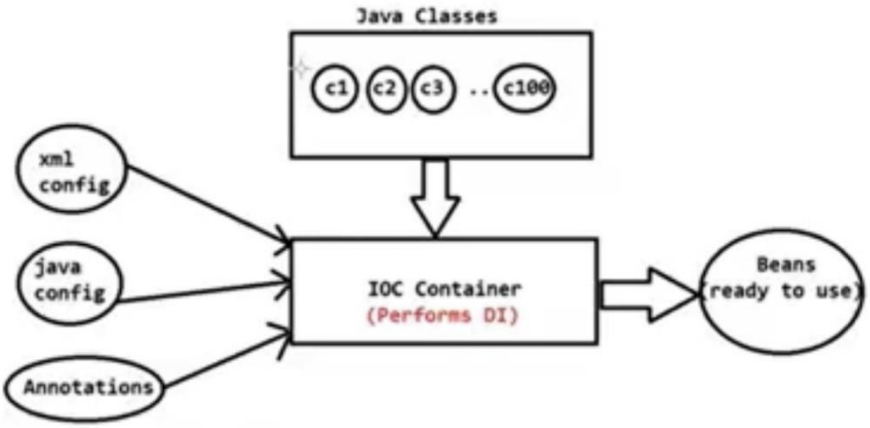
1. singleton

2. prototype
3. request
4. session

Note: For singleton beans objects will be created when IOC starts

- > When we call context.getBean(.) method then object will be created for prototype beans
- > For prototype beans everytime new object will be create

Diag-9



Imp dimag -10

by`name dependency injection

=> Spring IOC supports Autowiring concept that means Spring IOC having capability to identify the dependent and inject dependent into target

=> Autowiring having mode

1. byName
2. by Type
3. constructor
4. no

-> **byName** means if target class variable name matched with any bean id/name in bean configuration file then IOC will consider that as dependent bean and it will inject that dependent bean object into target object.

=> **byType** means it will check data type of the variable. With Datatype of variable if any bean class is configured then it will identify that as dependent and it will inject into target.

Note: We can configure one class for multiple times with different ids then we will get ambiguity problem in by Type scenario.

<bean id="xyz" class="in.bytecode.beans. DieselEngine" I>
<!- <bean id="abc" class="in.bytecode.beans.DieselEngine" |> ->

<bean id="car" class="in.bytecode.beans.Car" autowire="by Type">
</bean>

-> In by Type mechanism if we have more than one bean matching with type then we will get ambiguity problem.

=> To overcome ambiguity problem we need to use **'autowire-candidate=false**

<bean id="xyz" class="in.byte code.beans.DieselEngine" autowire-candidate="false">

<bean id="abc" class="in.byte code.beans.DieselEngine" |>

<bean id="car" class="in.byte code.beans.Car" autowire="by Type"> </bean>

Note: When we configure autowiring with "byName" or "by Type" it is performing setter injection by default

-> In by Type mechanism if we have more than one bean matching with type then we will get ambiguity problem.

=> To overcome ambiguity problem we need to use 'autowire-candidate=false
<bean id="xyz" class="in.bytecode.beans.DieselEngine" autowire-candidate="false"|>

<bean id="abc" class="in.bytecode.beans.DieselEngine" />

<bean id="car" class="in.bytecode.beans.Car" autowire="by Type"> </bean>

Note: When we configure autowiring with "byName" or "by Type" it is performing setter injection by default and setter method is mandatory.

=> If we want to perfom Autowiring through constructor then we can use 'constructor' mode I

<bean id="xyz" class=*in.bytecode.beans. DieselEngine" autowire-candidate="false"/>

<bean id="abc" class="in.bytecode.beans. DieselEngine" />
<bean id="car" class="in.bytecode.beans.Car" autowire="constructor"/>

What is Spring Framework ?
Spring Modules
Strategy Design Pattern
Spring Core Introduction
10C Container
Dependency Injection
Setter Injection
Constructor Injection
Field Injection <propery/> tag
15)
<constructor-arg/> tag
Bean Scopes

- singleton
- prototype <ref /> attribute

Autowiring

- by Name
- by Type
- Constructor