

Spring Framework : Popular frameworks to develop java application

Highly Modular in nature

Framework :

1. Strict and disciplined implementation of an architecture
2. Reduce the boiler-plate code
3. Abstract implementations of API
4. Focus more on Business logic

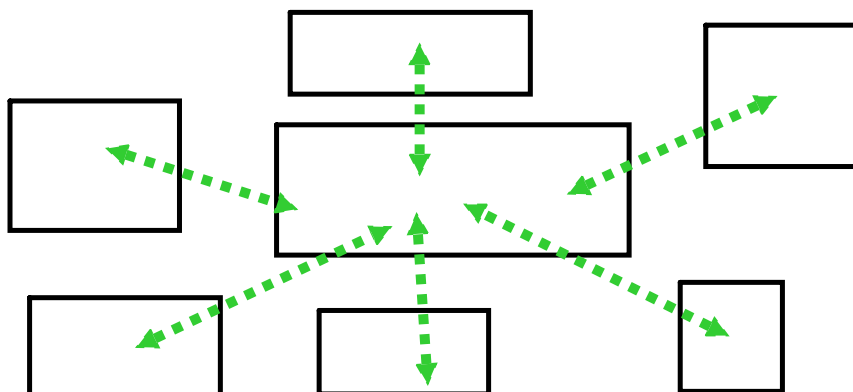
**J2EE Framework :****1. Complex in nature :**

Service : need to create lots of interface, abstract classes, inherit class and interface  
reduces the productivity of developer  
reduces the efficiency  
Uses lots of deployment descriptors : xml files

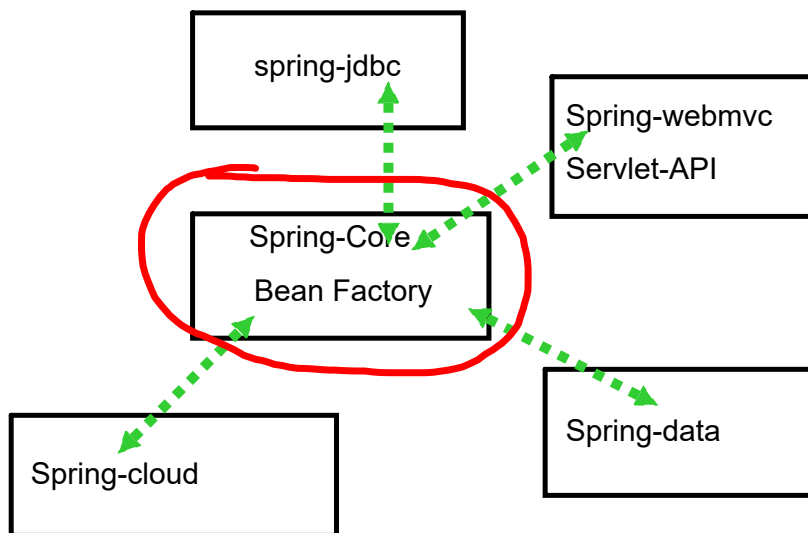
**Rod Johnson**

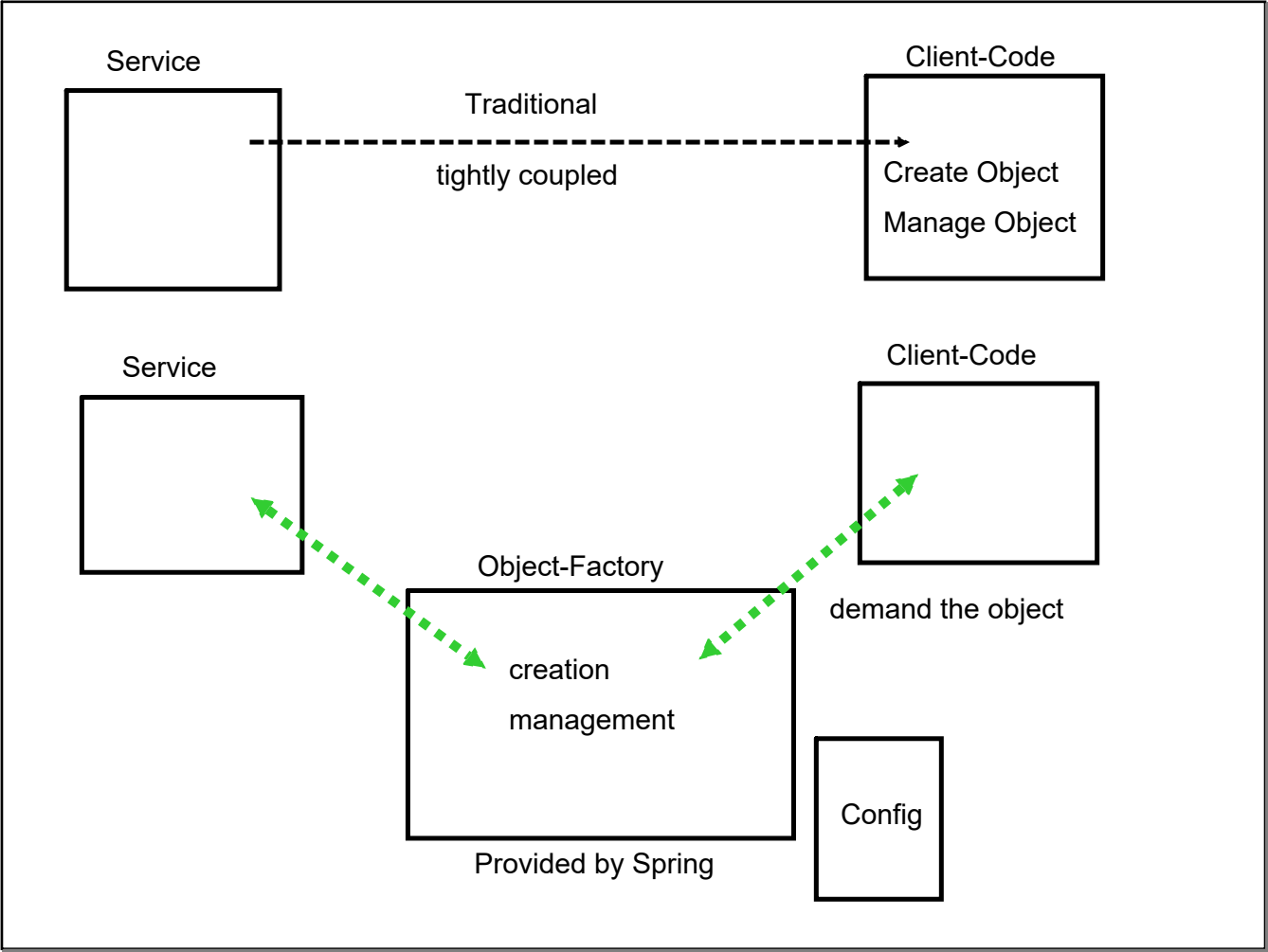
Create a tool/resources : Object Factory/Bean Factory  
create and manage object

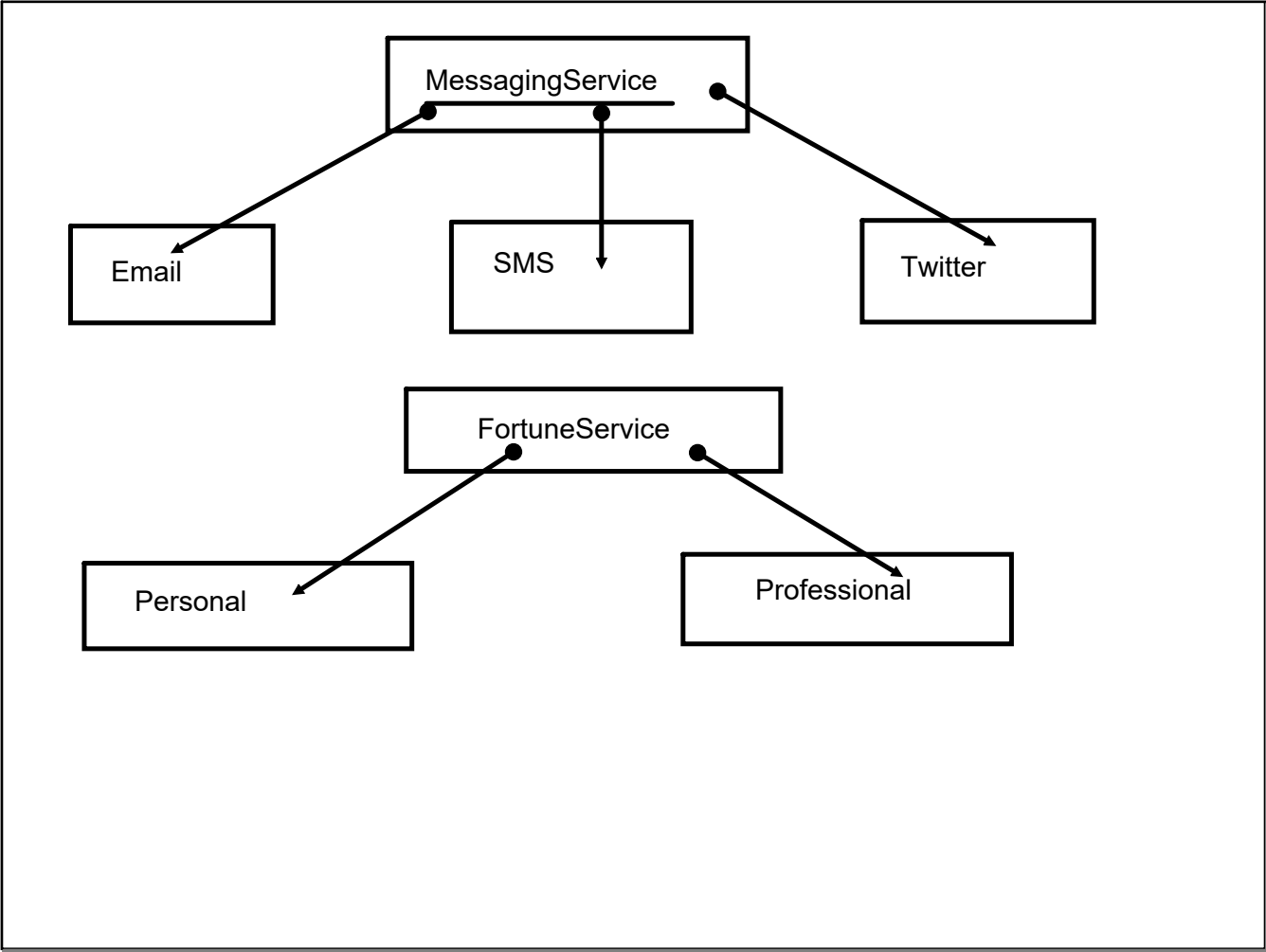
Spring : Lightweight

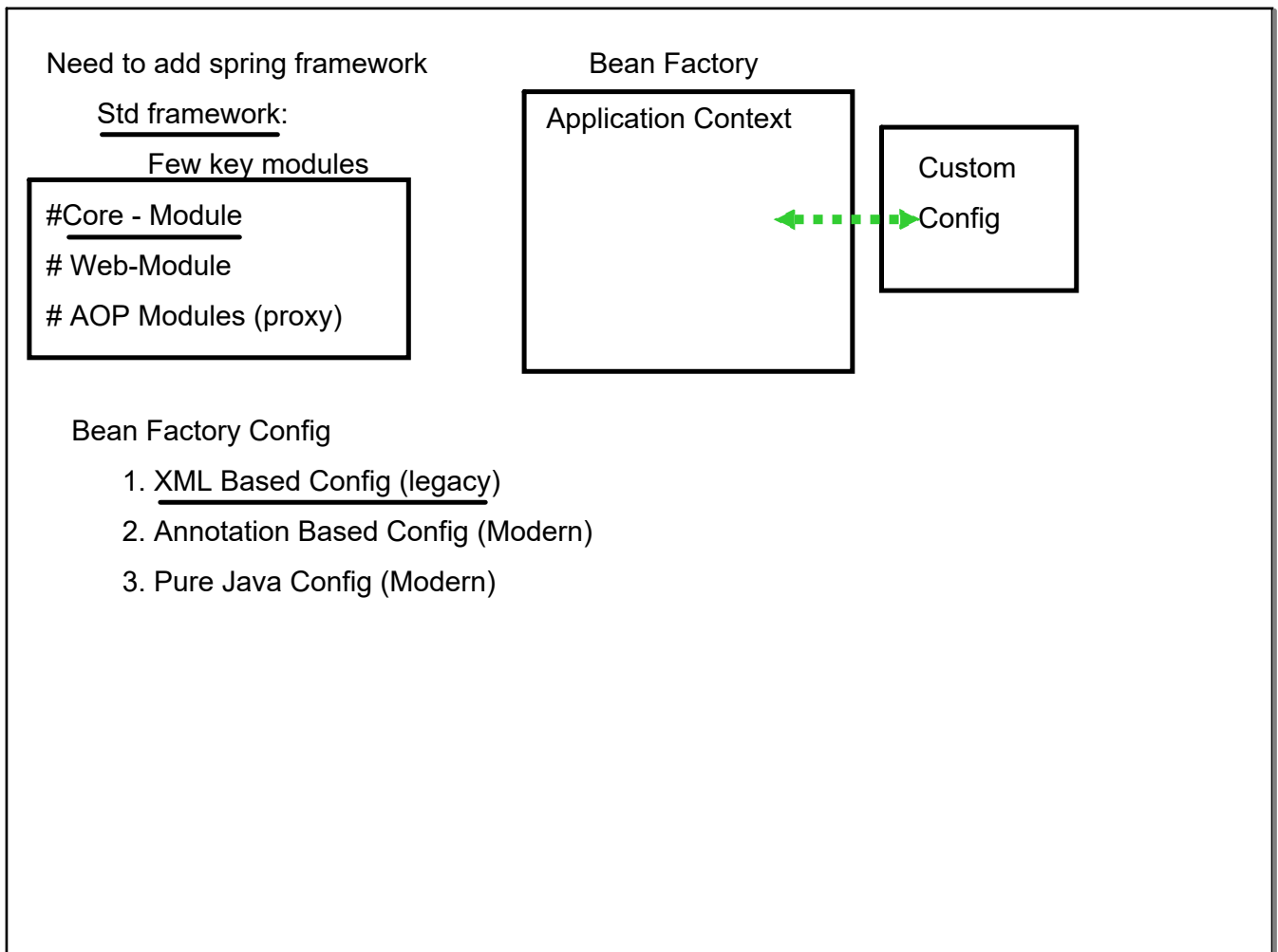


1. All implementation will be based on Object/Bean Factory
2. highly modular in nature
3. All implementation will be based on POJOs









XML Based config : XML file + with spring dependency add (additional tags)  
xml config file

BEANS : Container(Object/Bean Factory) Managed Object

Two key principals of Bean Factory

1. IoC : Inversion of Control
2. DI : Dependency Injection

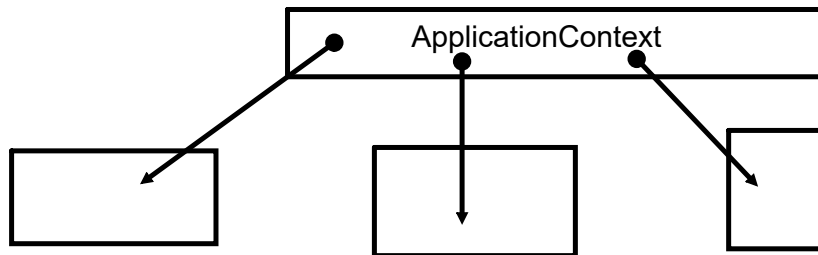
IoC : Outsourcing the (control of )creation and management of Object

Bean Factory :

to represent multiple classes

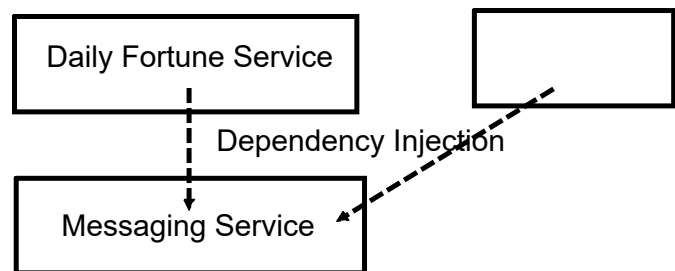
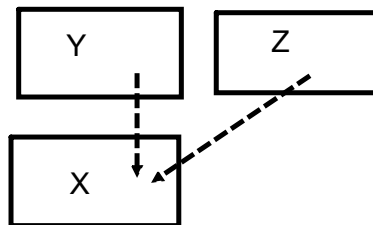
# type of config (xml, java...)

# env (java, web ...)





Dependency

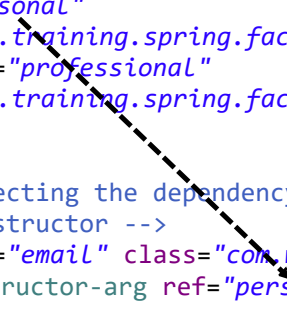


Two types of Dependency injection

1. Constructor based
2. Setter based

```
<bean id="personal"
class="com.wf.training.spring.factory.service.PersonalFortune"></bean>
  <bean id="professional"
class="com.wf.training.spring.factory.service.ProfessionallFortune"></bean>

  <!-- Injecting the dependency : How -->
  <!-- Constructor -->
  <bean id="email" class="com.wf.training.spring.factory.service.EmailService">
    <constructor-arg ref="personal"/>
  </bean>
```



### Injecting Literal Values :

Delegate values to a text file : properties file  
key-value pair

### IoC | DI : Create a Bean

#### Managing the Bean

1. Scope
2. Life cycle

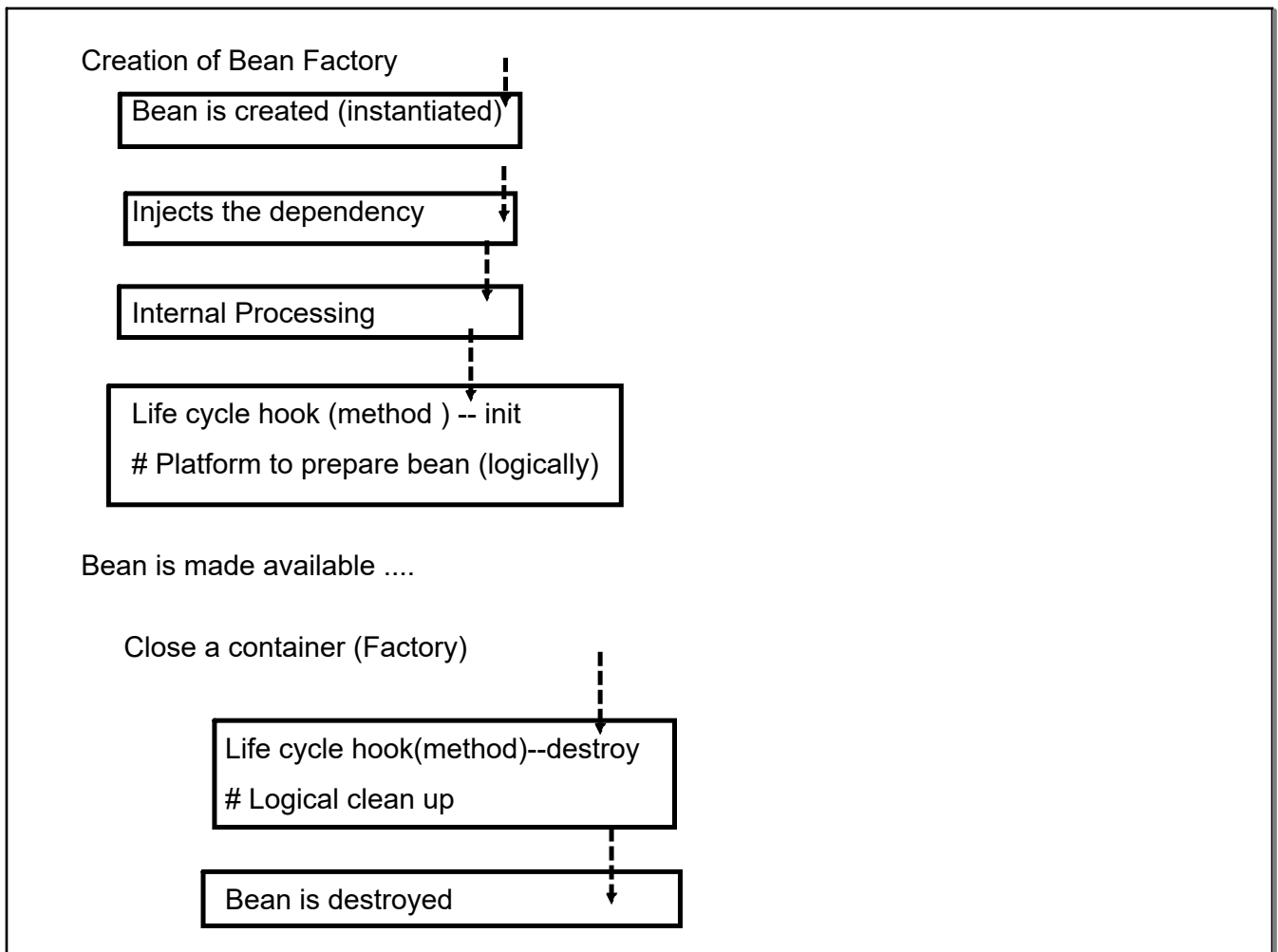
### Scope : Accessibility of Bean

by-default : singleton  
explicit : prototype

request

session : Web App

application



Prototype beans : Bean Factory does not manage the complete life-cycle

Annotation based configuration

still xml file : referencing the path/location

want to create and expose a bean of Messaging service

@Component : Any class decorated this ann. will instantiated by Bean Factory

XML : need to mention package(s) to scan for @Component ann.

id : Class Name is by-default considered as id : first character in lower case

DI ways:

1. Constructor
2. Setter Based
3. Field Based

@Autowired : searches for bean of param type, if found , inject it

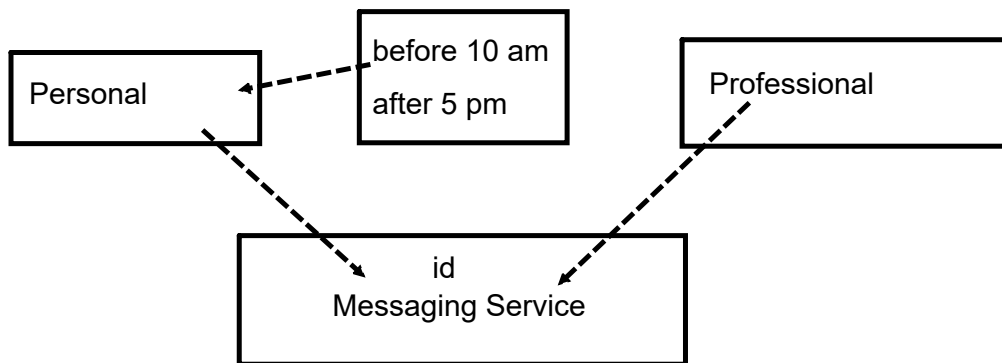
@Qualifier : differentiate the bean

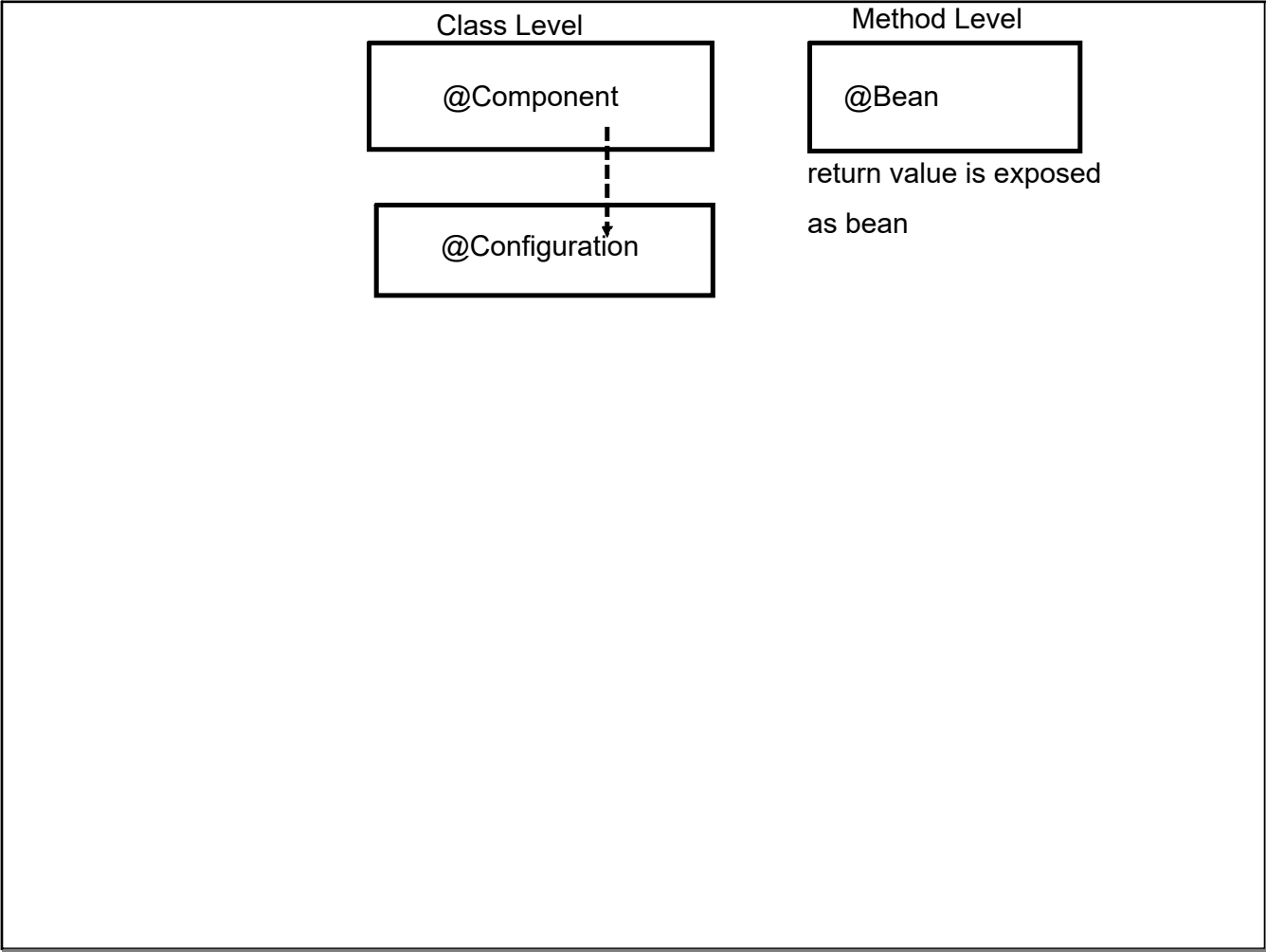
Literal Value : @Value

@Scope : for defining the scope of the bean

Pure Java Config

XML file will be replace with a Java class



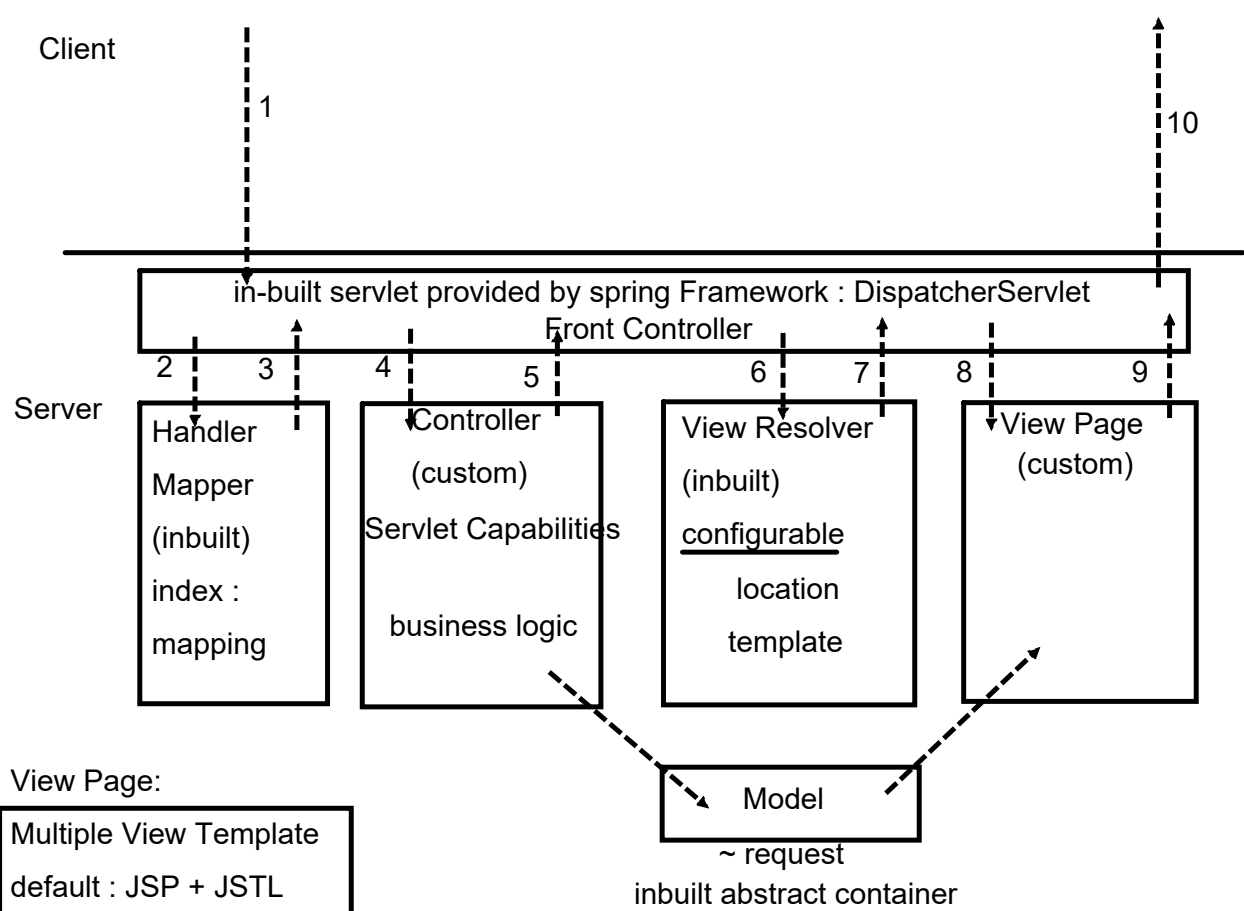




Spring Web-MVC Module : implements MVC architecture strictly

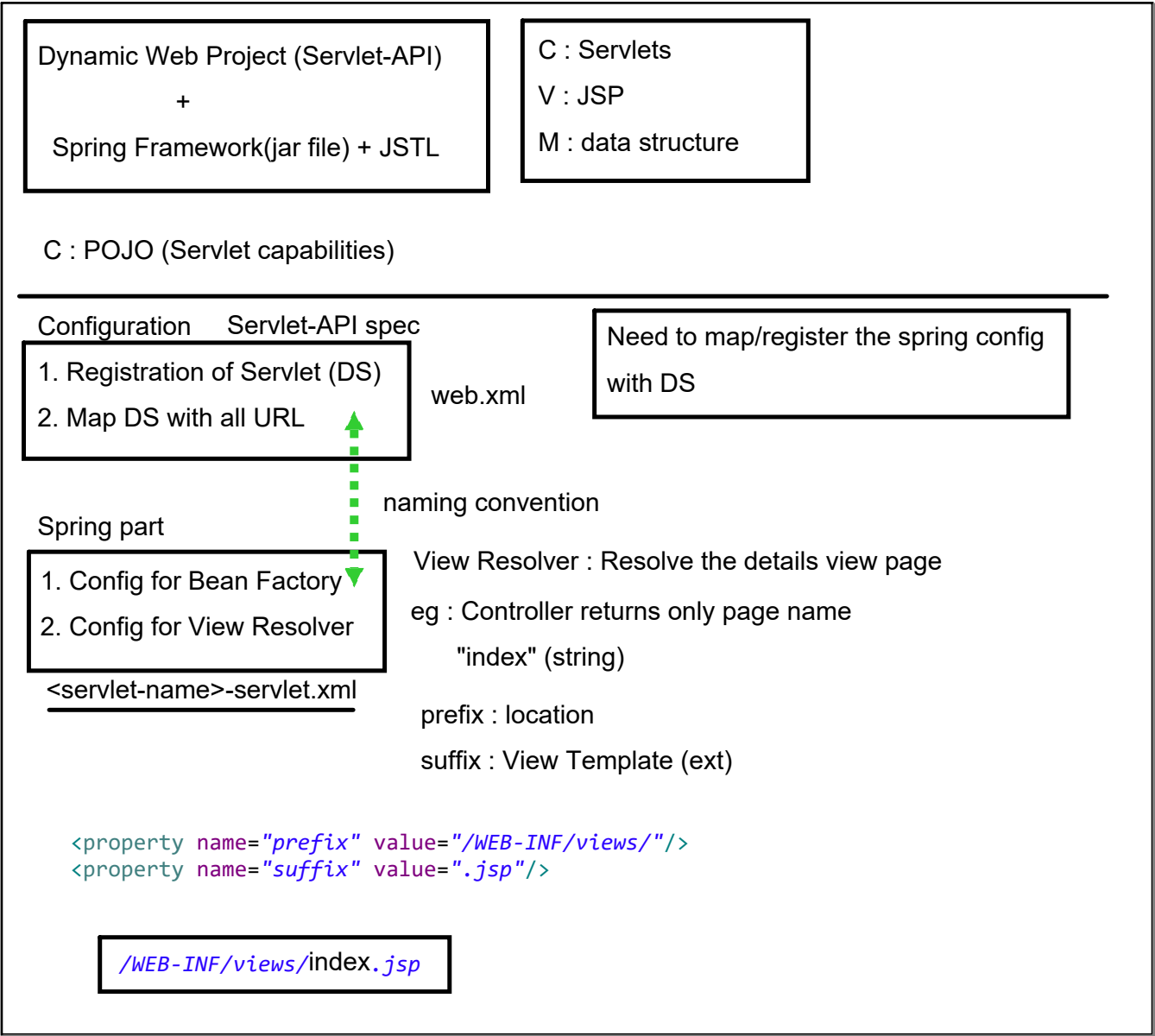
integrate the Servlet API and implement it in MVC

Front Controller Design Pattern

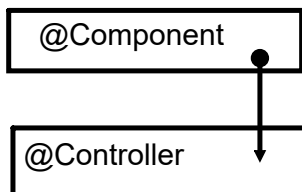


View Page:

Multiple View Template  
default : JSP + JSTL  
Thymeleaf  
Mustache  
Velocity  
Tiles  
FreeMarker



Controller : POJO, registered with HAndler Mapper



All handler methods must have unique URL mappings

Mapping of input field with java class : getter/setter

Mapping is flexible

Maven Project implementation

1. Create Maven Project
2. web-archetype
3. add the Server Runtime Library (Tomcat)
4. Add java 1.8 support
5. Add dependency:
  1. spring framework dependency
  2. servlet + jsp + jstl

Copied sources and xml files

Convert all config to pure Java

Servlet based config : web.xml

Spring based config : dispatcher-servlet.xml

~ Java Classes

Java class

web.xml ( de-facto std ) : without web.xml project will not be packaged (war)

~ plugin is provided by maven to build and package

Java class for servlet based config (web.xml)

- # register the DS : inbuilt class for DS register ( inherit this class)

- # mapped the url (all url targeted to DS)

Java Class for Spring based config (dispatcher-servlet.xml)

- # component scanning path

- # expose the bean of type ViewResolver

relate the spring config with servlet config

Handling the forms : critical :

Spring forms : Custom Tag Library : need to add support of taglib  
allows to do mapping of JAVa class with HTML forms,  
able to control behavior of UI/forms through java classes

Introducing Validation :

defining the validation rules in java class, form will follow it

Validation rules : Validation API

Hibernate Validator API : add dependency in pom.xml

Expose Validation annotations

javax

hibernate

Validation : Standard API for JAVA classes :  
Java Validation API ( javax.validation.constraints)

Hibernate Validation API ?

Java Validation API : spec : set of rules : set of interface

@NotBlank

new impl.

Hibernate Validation API (implementation)

@NotBlank

javax : preferable : prevent Vendor locking

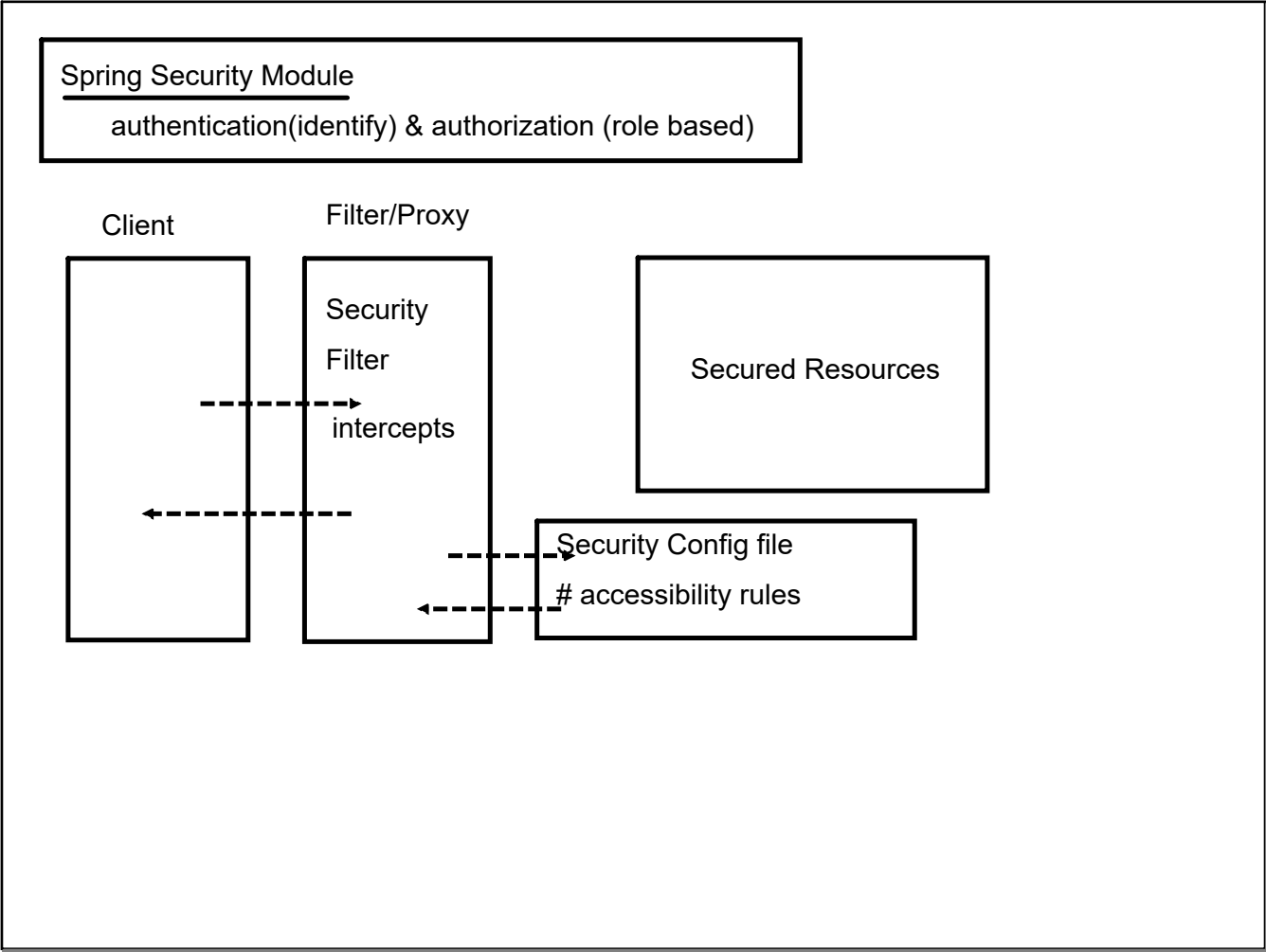


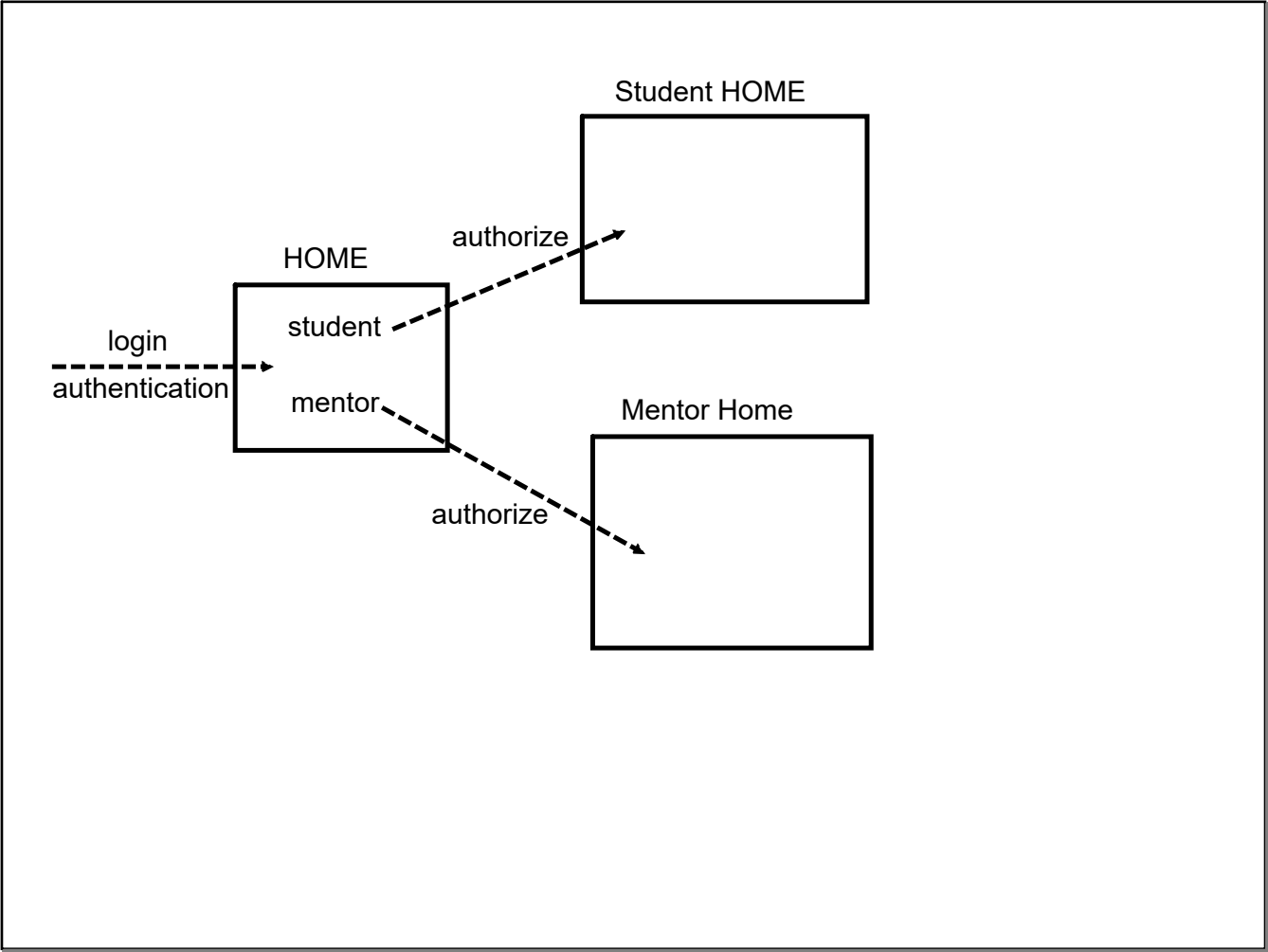
### Validation

Client-Side Validation : HTML5 + JS

Server-Side Validation : Java | need to add extra code

### Custom Validation Annotation





absolute URL : context-path/

Every JSP has a pre-declared object to access context path

1. Dependency : Security Module ( multiple sub-api)
    1. spring-security-web (security filter)
    2. spring-security-config (custom config)
    3. spring-security-taglibs (tags library)
  2. activate/initialize the security filter : java class
  3. add custom config : java class  
inherit a inbuilt default config class, we override as per our requirement
- Default config : all resources are by default secured  
: form based authentication ( inbuilt login form )

### Spring security

#### 3 type of authentication

1. HttpBasic : not recommended
2. form based auth (inbuilt login)
3. form based auth (custom login)

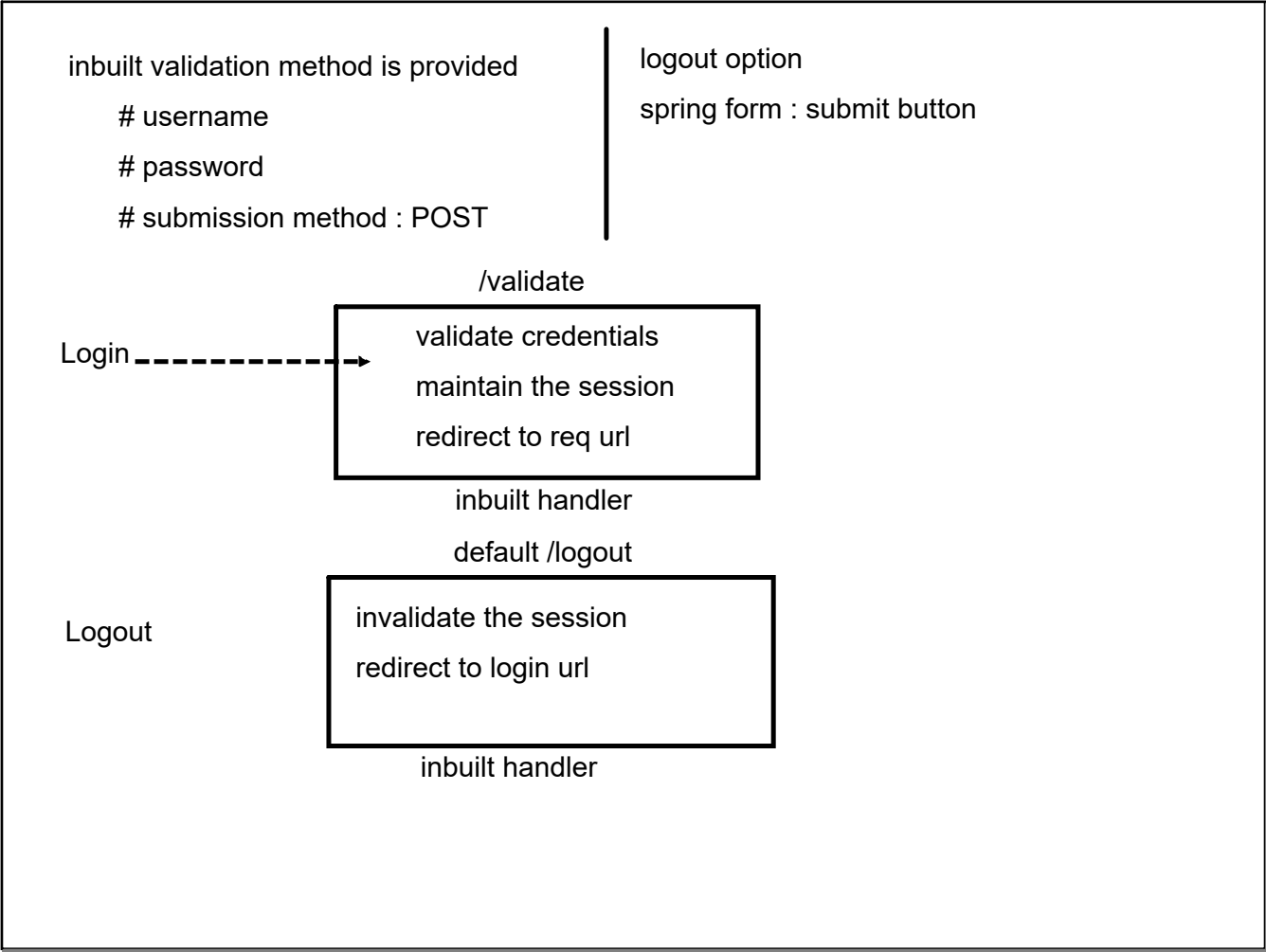
#### # spring security filter :

Every form must be spring form  
plain html forms: submission will be blocked

### Spring forms :

security : prevent CSRF attack  
cross site request frogery

security token :  
generated by website  
need to be submitted back



Spring-Boot : Framework designed on top of spring  
make the development of spring application easier

- => API dependency management
- => Auto Configuration
- => Reduction in boiler-plate code
- => Create independent self-sufficient application

API dependency management

curated list of relevant dependencies

std web app : all apis needed to support web application development

Auto Configuration

- => dependency you add in pom.xml : std-default config added
- => Specialized Annotation : adds config

Reduction in boilerplate code

=> custom configuration : properties files  
key-value (pre-defined : possible value)

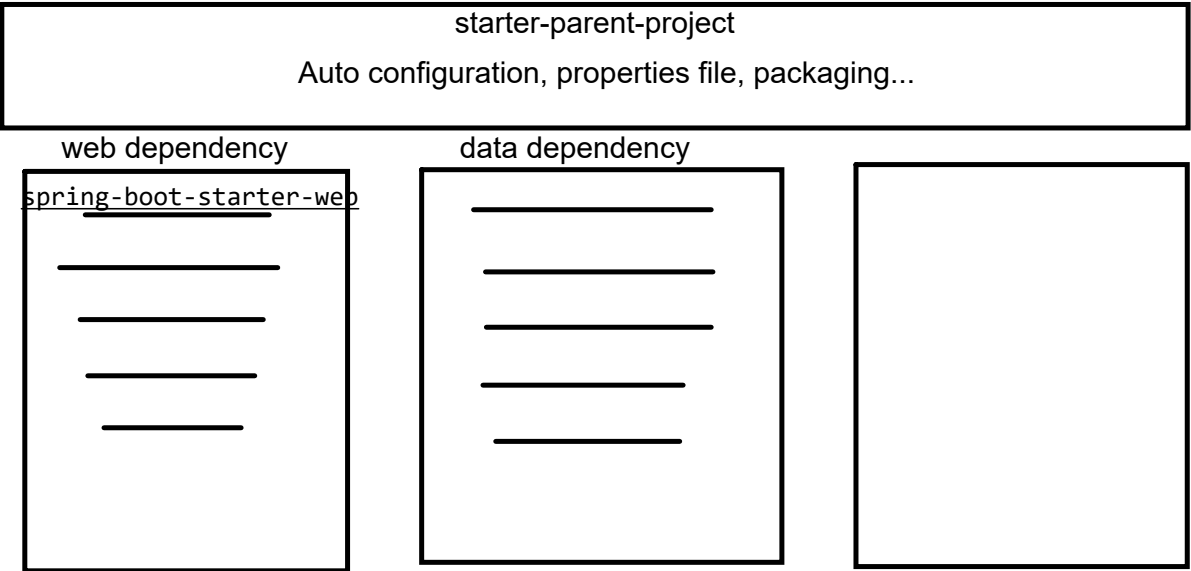
Self-Sufficient

- # Spring-boot application does not uses any support from IDE
- # maven : does not need any support of maven on developer machine
- # Embedded Tomcat
- # jar file : build,package,run from command terminal using java command



Creating a spring-boot app:

- 1. web-portal : create a basic spring-boot application, download, import
- 2. spring boot plugin

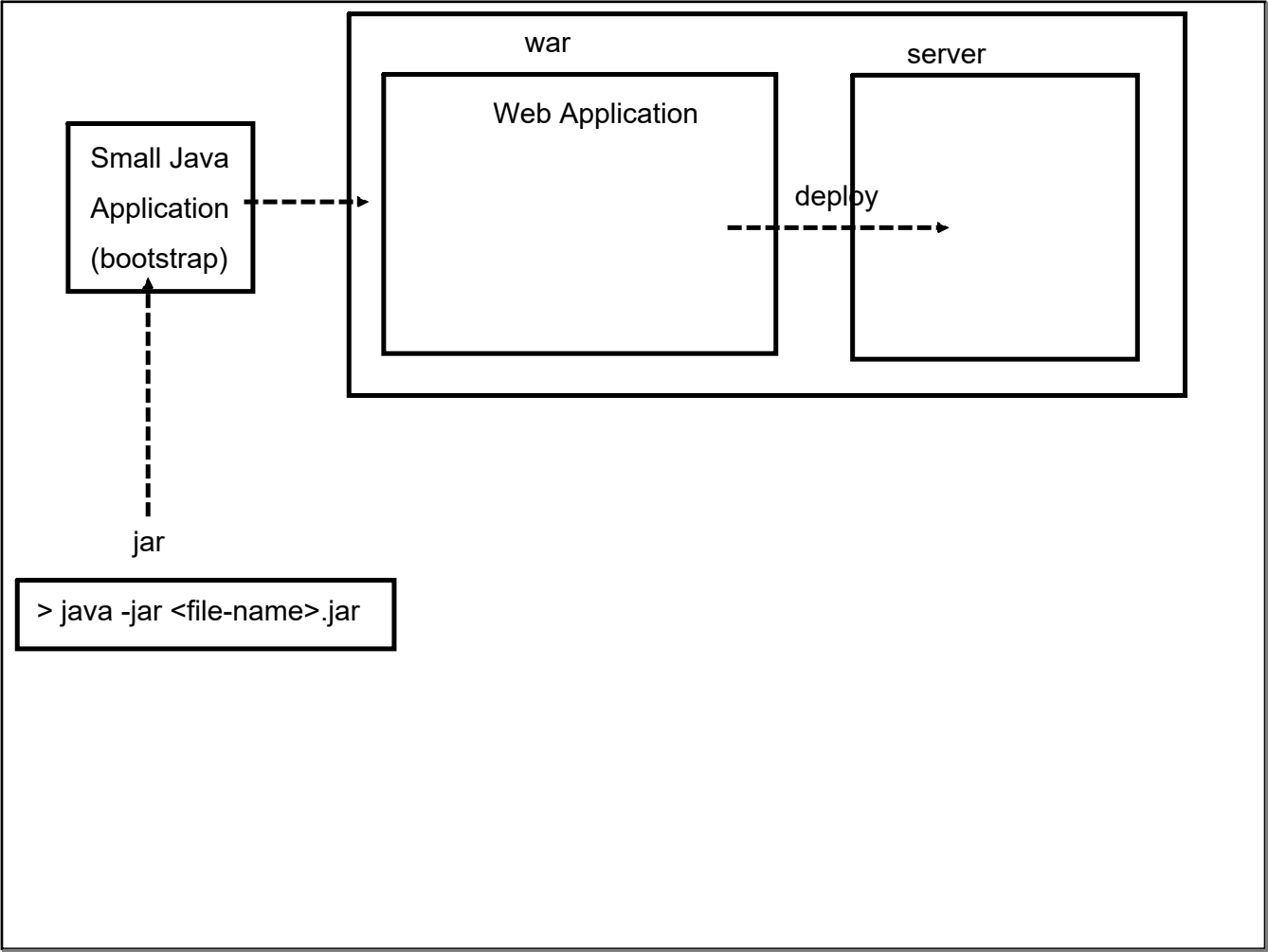


webapp : jsp+jstl view template

spring-boot : by default configured for thymeleaf view-template

templates : home for view page

application.properties : key-value (custom config)



# Adding support of JSP-JSTL (add dependency)  
# create webapp folder structure  
# config the prefix and suffix in application.properties file

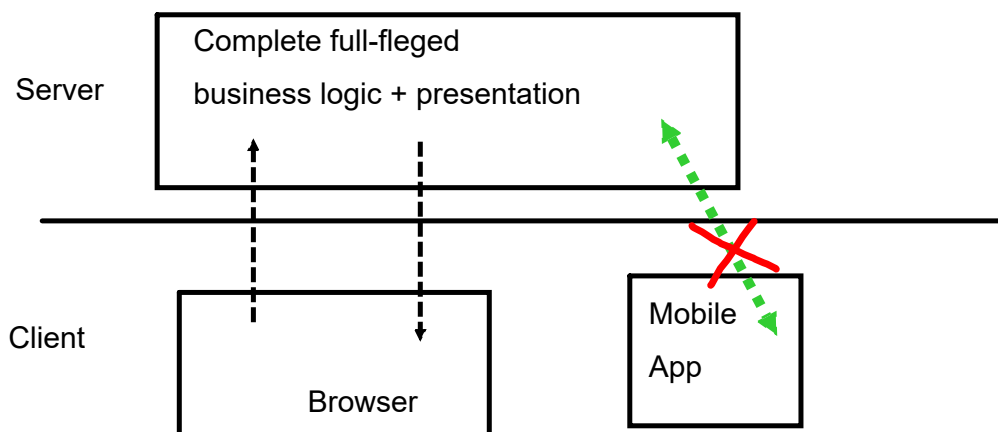
build using maven : CLI  
mvn <option> : if maven is installed  
mvnw <option> : if maven is not installed

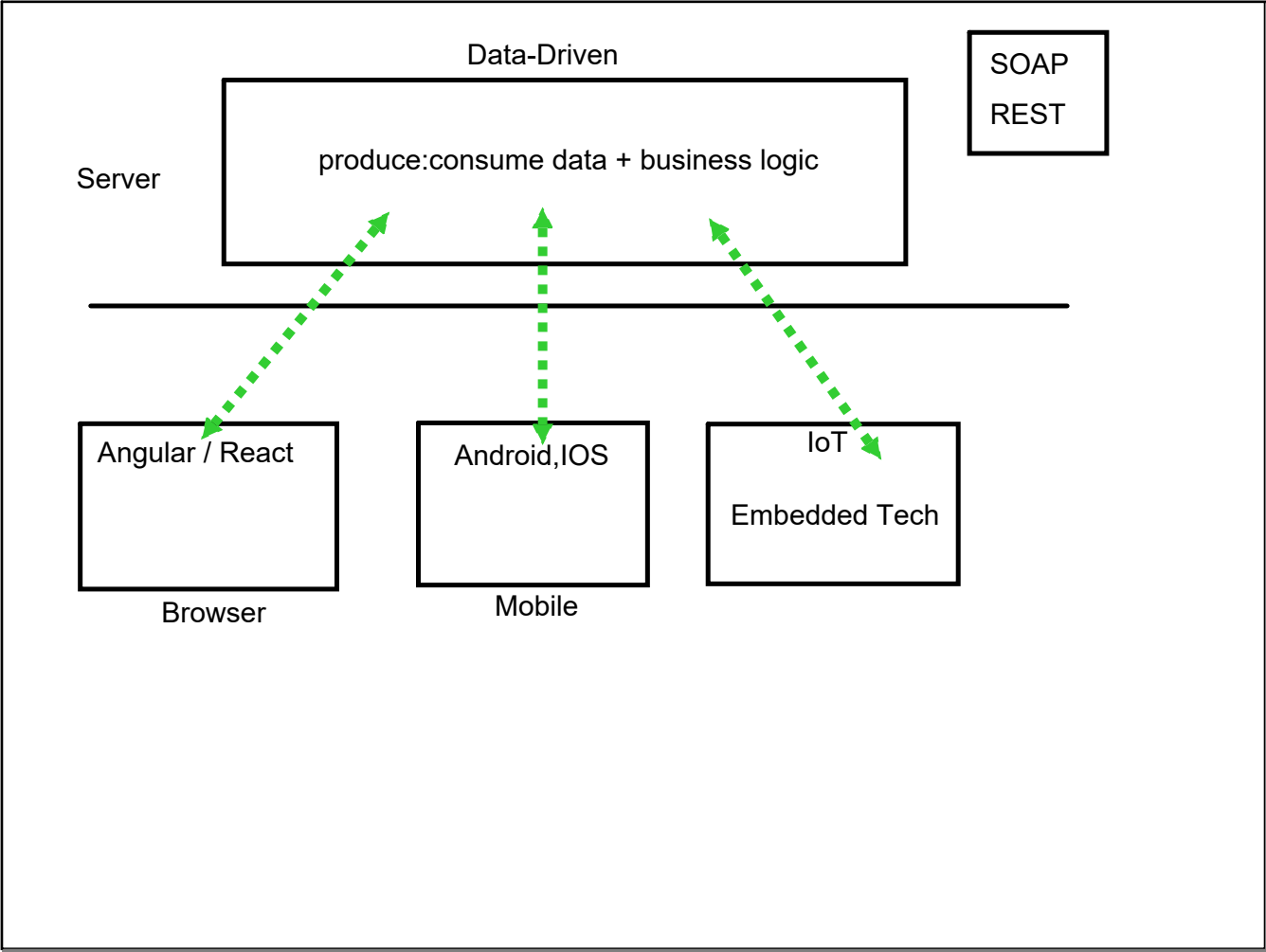
Converting Spring application to Spring boot

1. add dependency for spring security starter project : (activate the security filter)
2. add external dependency for security taglibs
3. add dependency for spring validator starter project
4. config : security config need to transferred
5. Transfer resources

### Creating a web application with REST Service

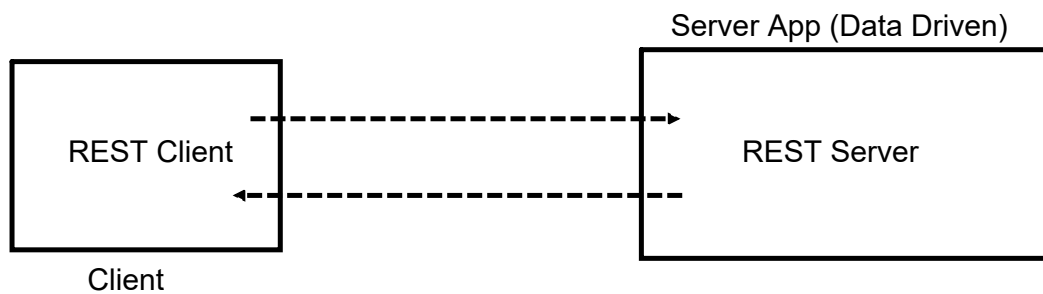
RESTful : Data-Driven Application





SOAP : Legacy :

Functional (API) : Object



1. How do i talk with Server App?

2. What is the data format ?

REST

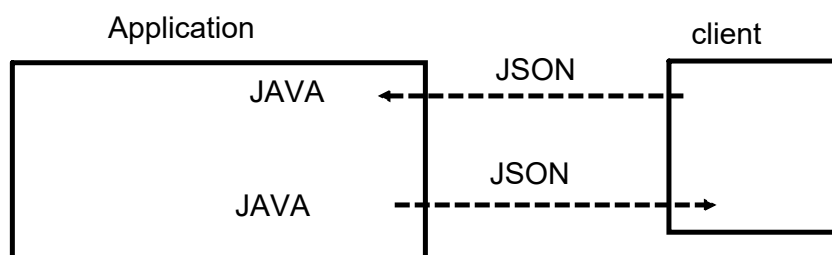
1. url based Interaction + HTTP Verb

2. text based : JSON(popular), XML, text, HTML



### REpresentation State Transfer

1. Stateless : not state maintainance ( client side responsibility)
2. Inherently not secured ( rely on backend framework)



Mapping JAVA<----->JSON

jackson-databind API

getter/setter

Create controllers that can work on REST Protocol

Use-Case

Employee : Restful application expose all crud functionality

add new record

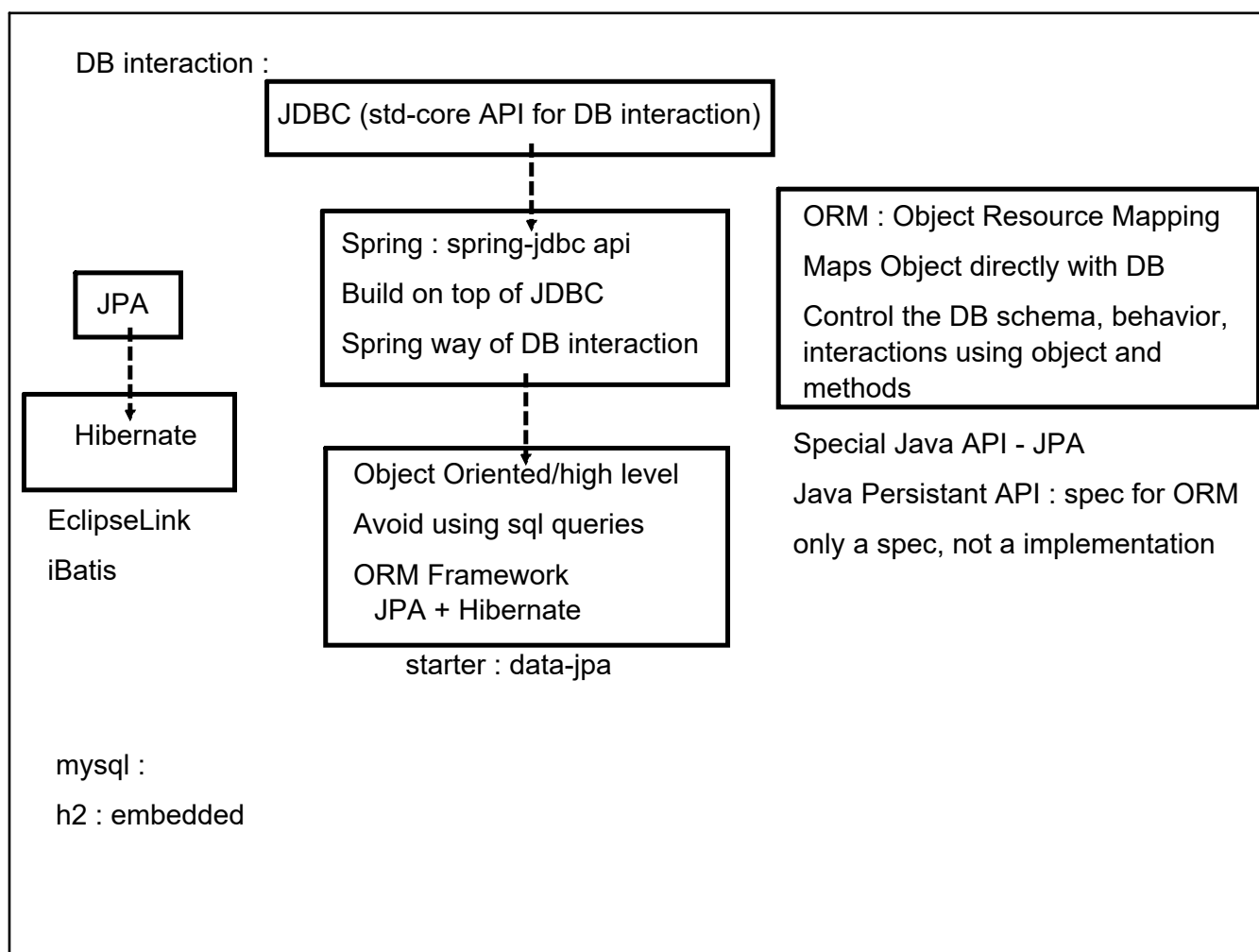
delete a record

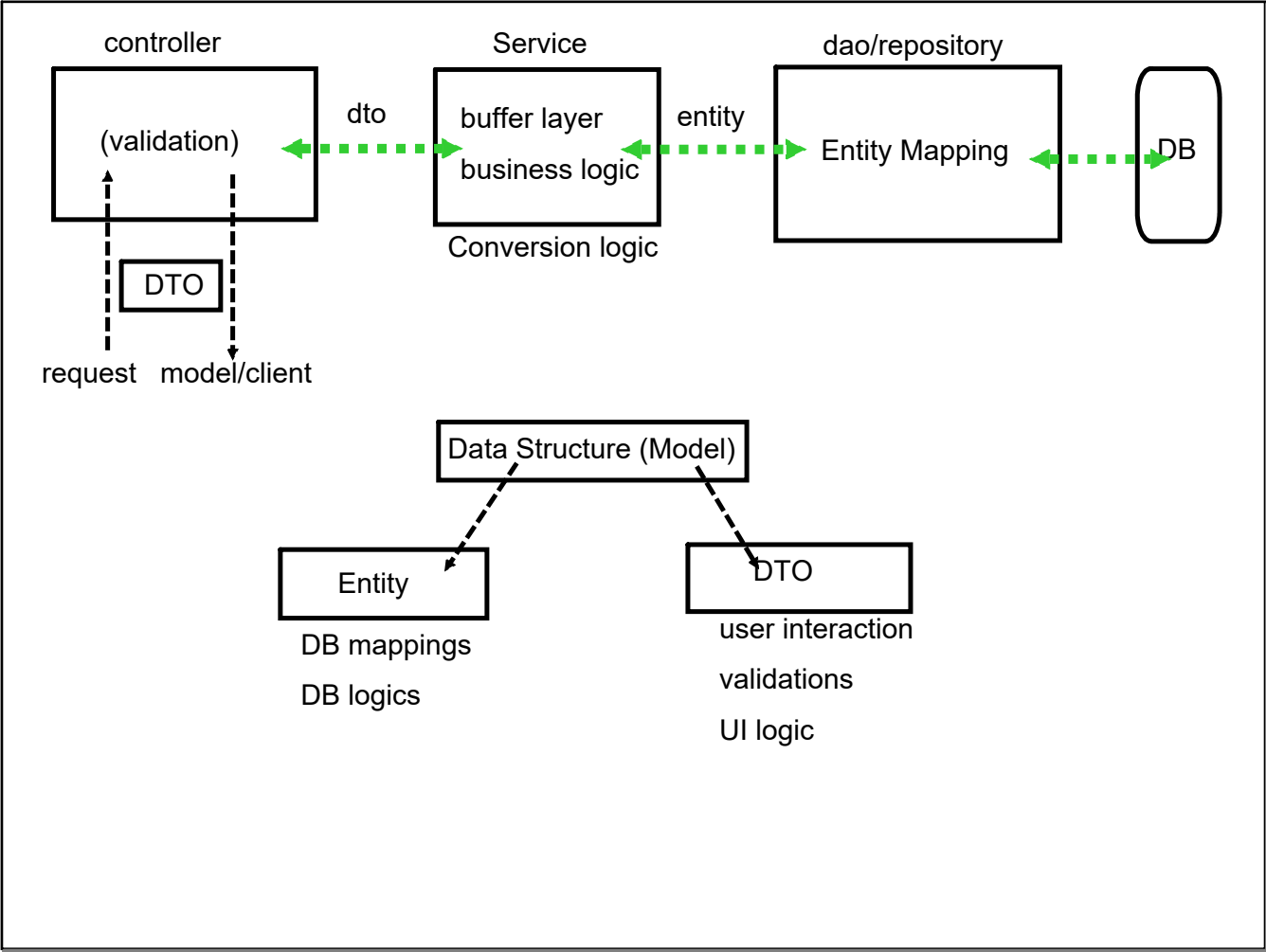
edit a record

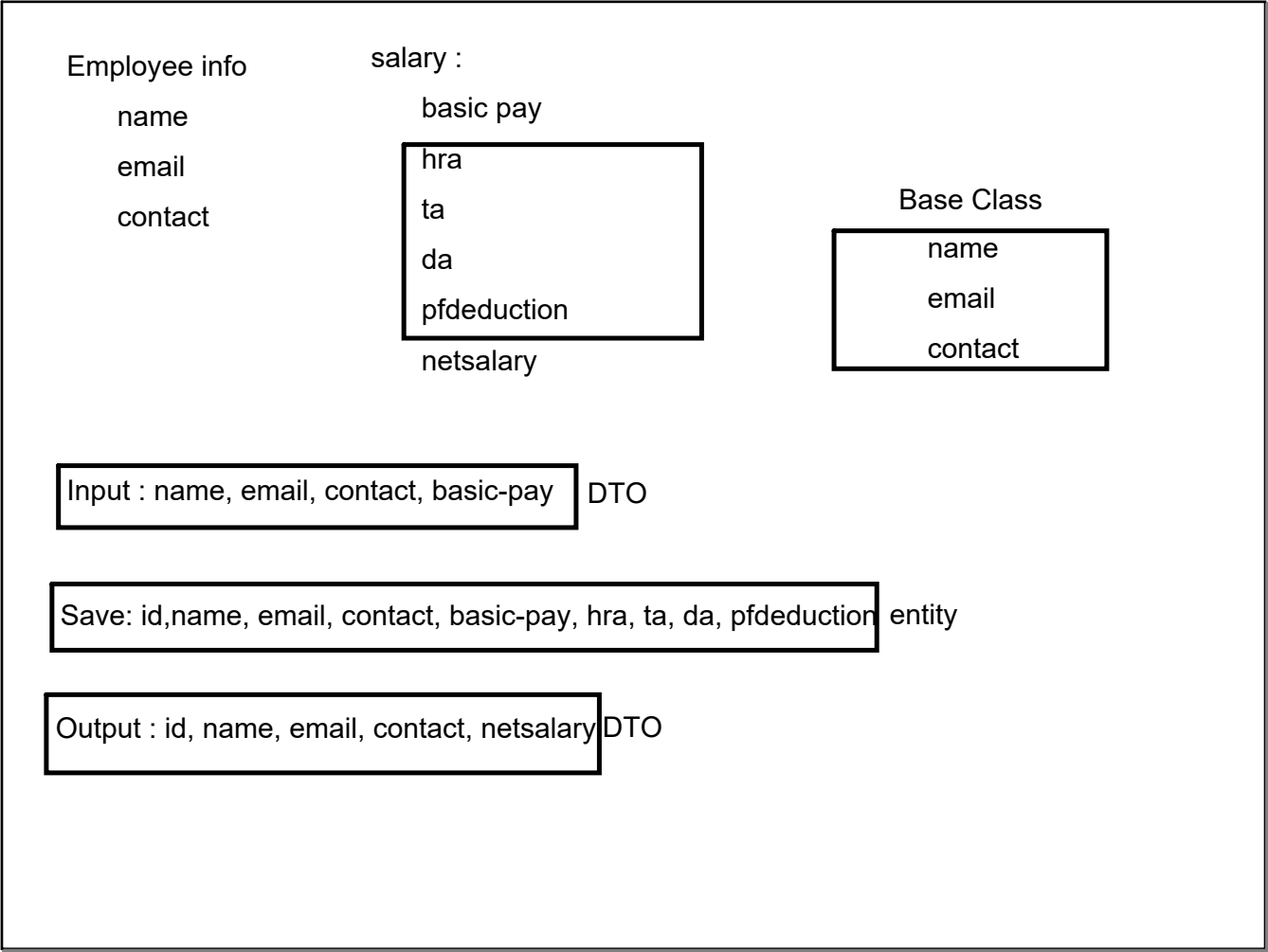
get all records

get a particular record

maintain data in database : Data module of spring







**Best Rest Practices:**

add : /addEmployee : /employees : POST

delete /deleteEmployee : /employees/{id} : DELETE

edit /editEmployee : /employees/{id} : PUT

fetch all : /employees : GET

fetch single : /employees/{id} : GET : /employees/4

<plural form of entity> : change the HTTP verbs