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

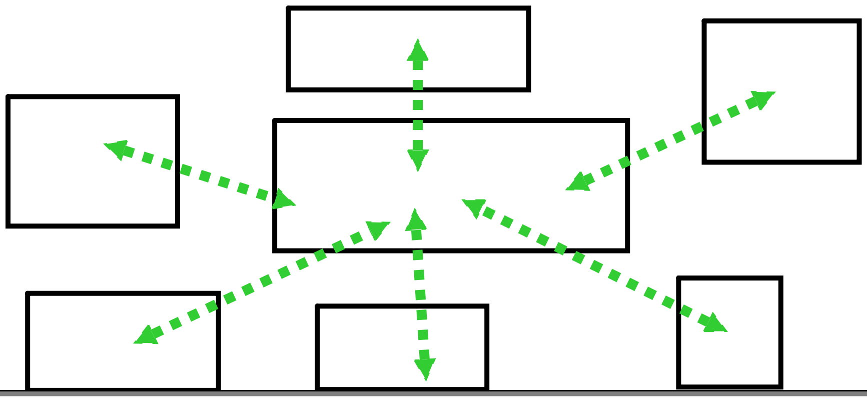
    Uses lots of deployment descriptors : xml files

Rod Johnson

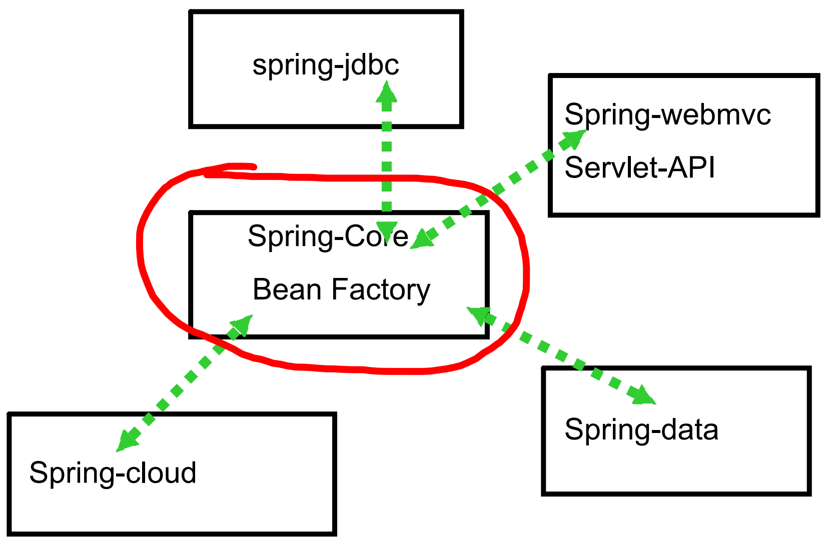
    Create a tool/resources : Object Factory/Bean Factory

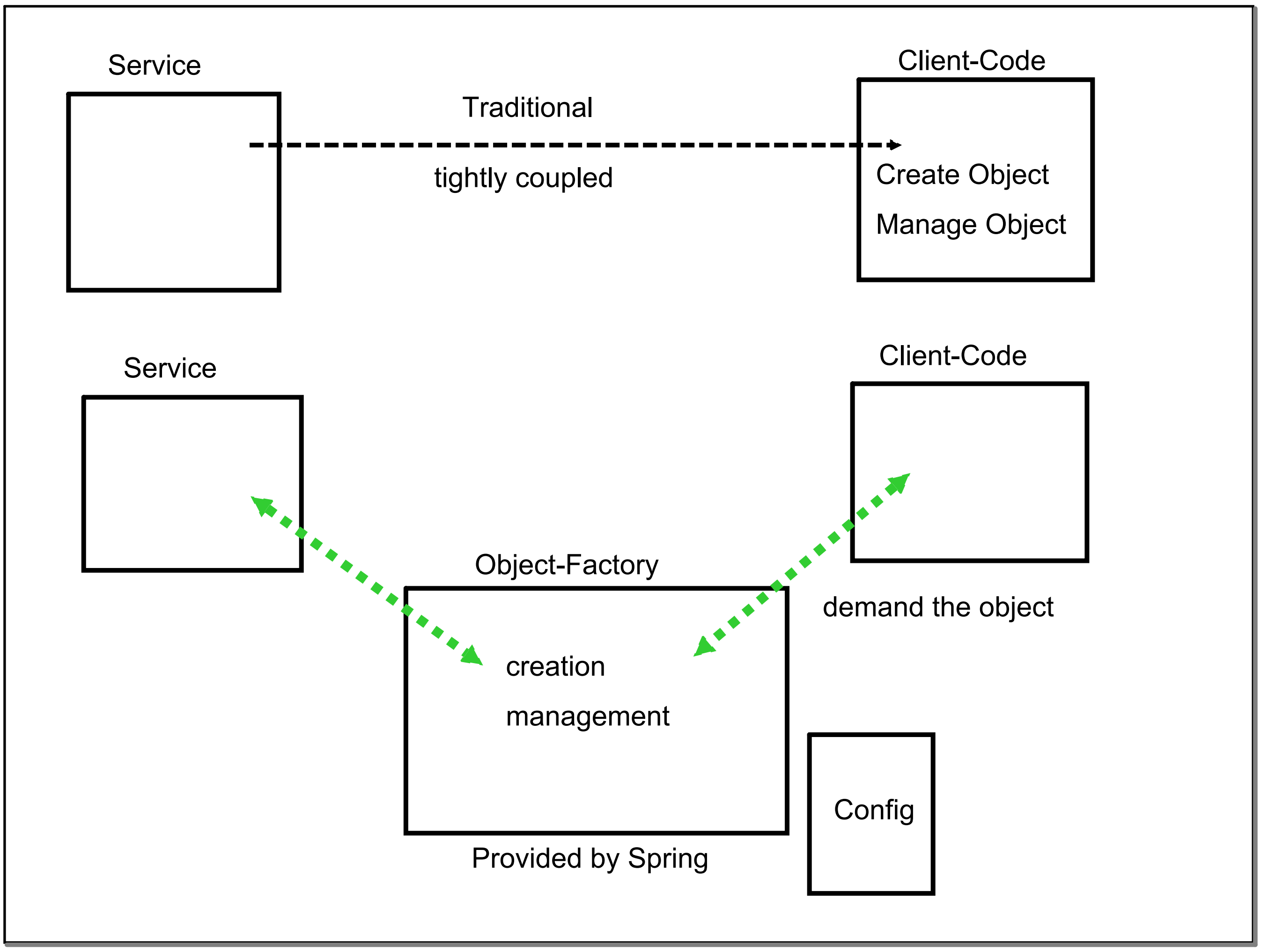
    create and manage object

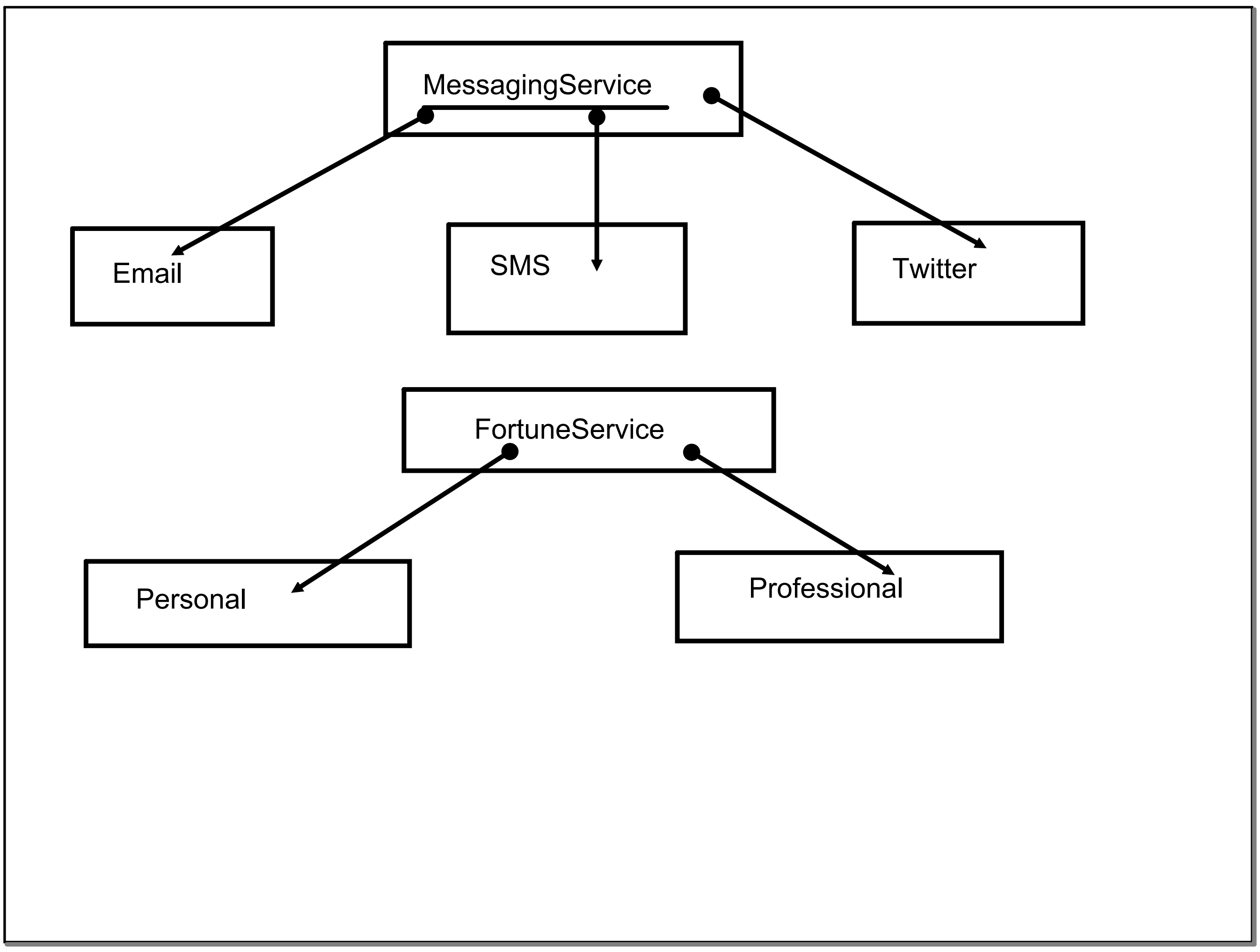
Spring : Lightweight



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




Service
Traditional
tightly coupled
Client-Code
Create Object
Manage Object
Service
Client-Code
Object-Factory
creation
management
demand the object
Provided by Spring
Config

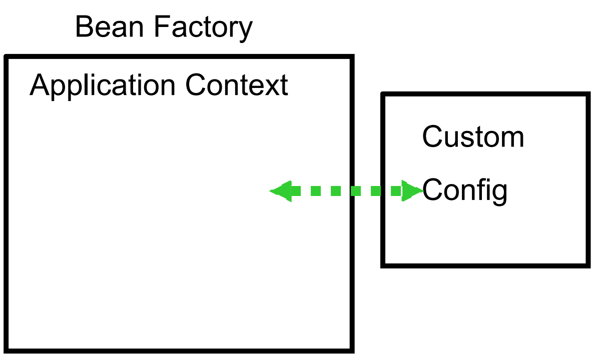

MessagingService
Email
SMS
Twitter
FortuneService
Personal
Professional

Need to add spring framework

Std framework:

Few key modules

- #Core - Module
- # Web-Module
- # AOP Modules (proxy)



Bean Factory Config

1. XML Based Config (legacy)
2. Annotation Based Config (Modern)
3. Pure Java Config (Modern)

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

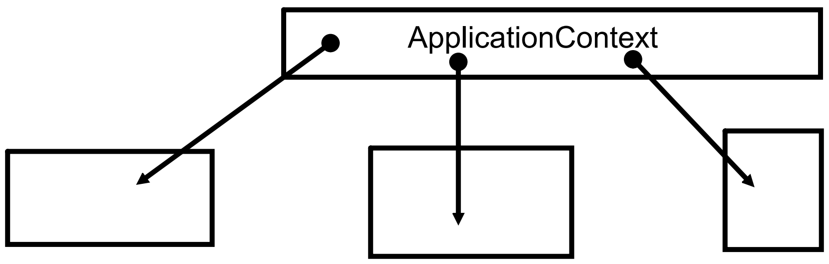
IoC : Outsourcing the (control of)creation and management of Object

Bean Factory :

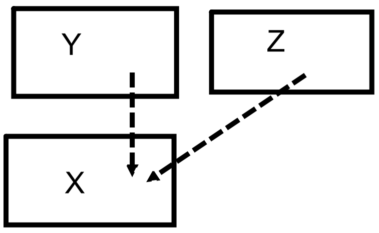
to represent multiple classes

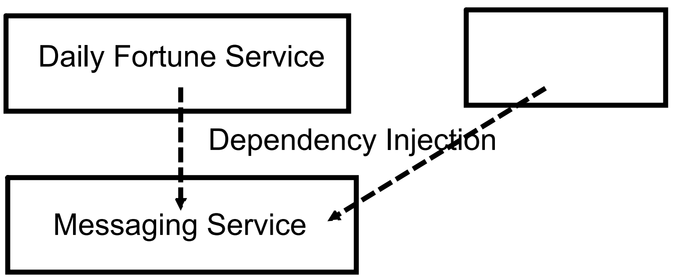
# type of config (xml, java...)

# env (java, web ...)



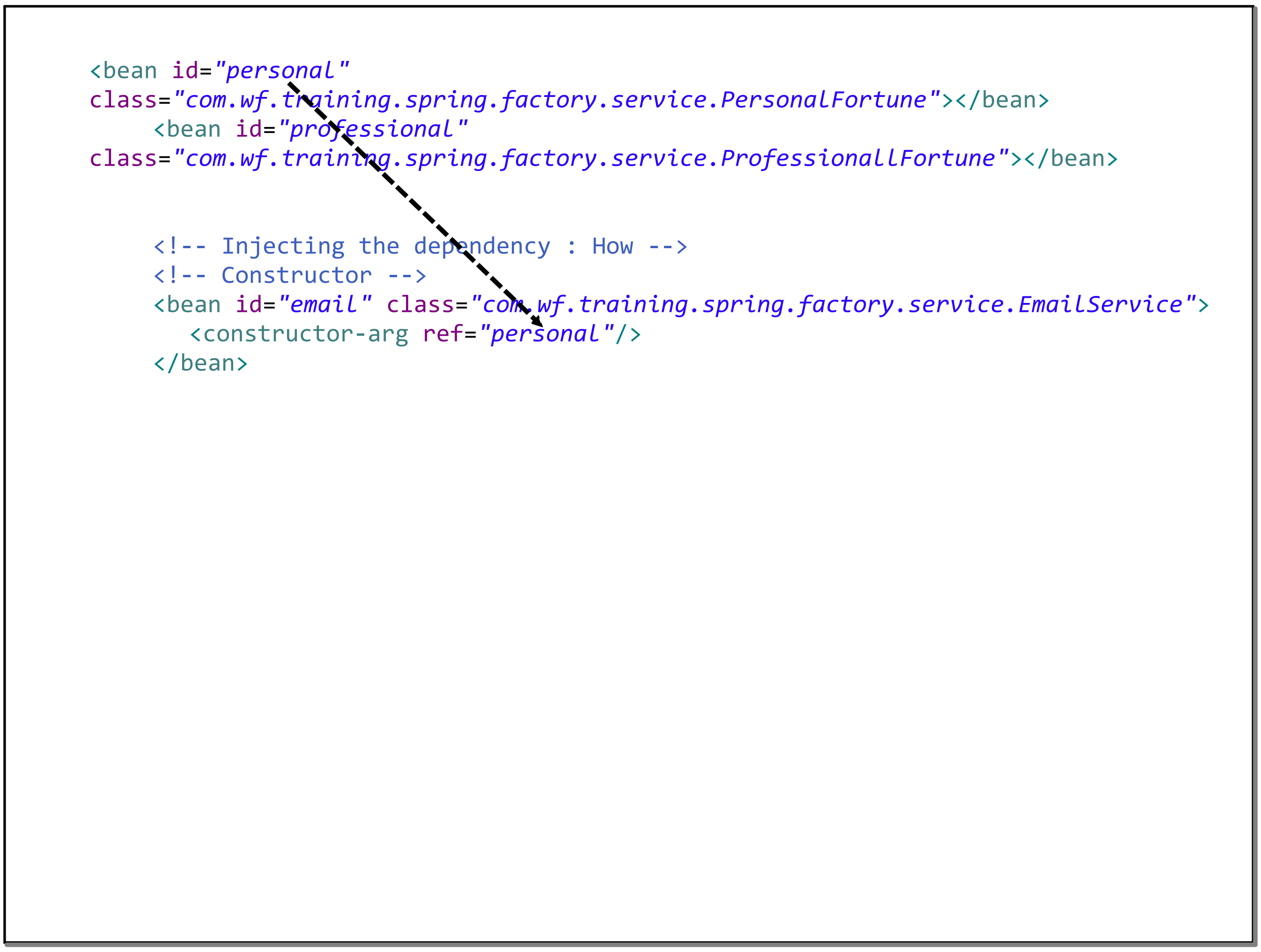
Dependency





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.ProfessionalFortune"></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

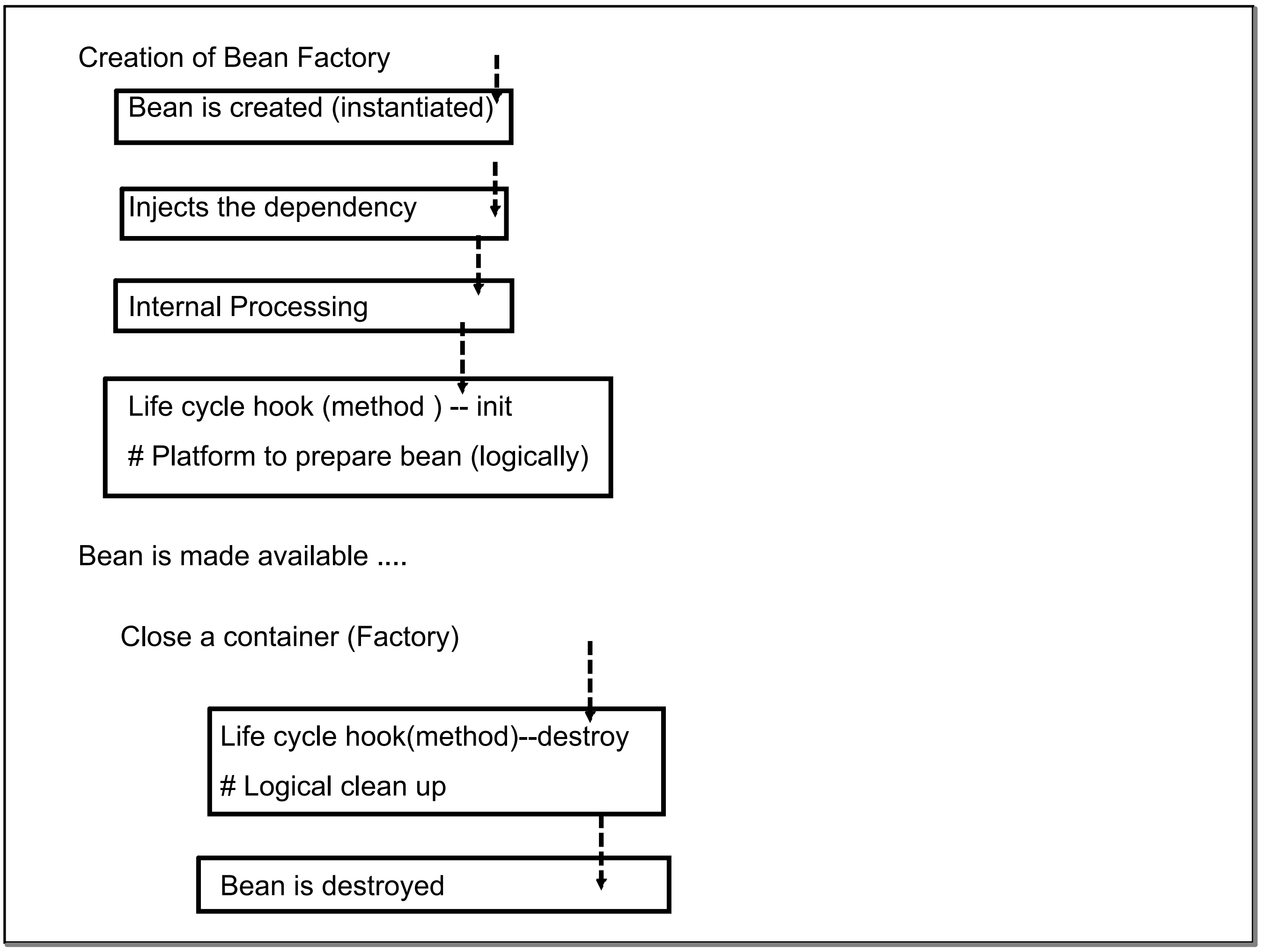
by-default : singleton

explicit : prototype

request

session : Web App

application


Creation of Bean Factory
Bean is created (instantiated)
Injects the dependency
Internal Processing
Life cycle hook (method) -- init
# Platform to prepare bean (logically)
Bean is made available ....
Close a container (Factory)
Life cycle hook(method)--destroy
# Logical clean up
Bean is destroyed

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 intantiated 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

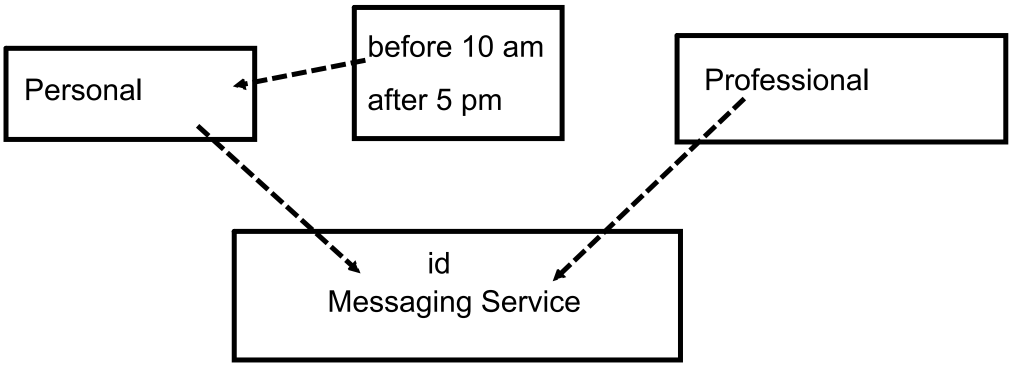
@Qualifier : differentiate the bean

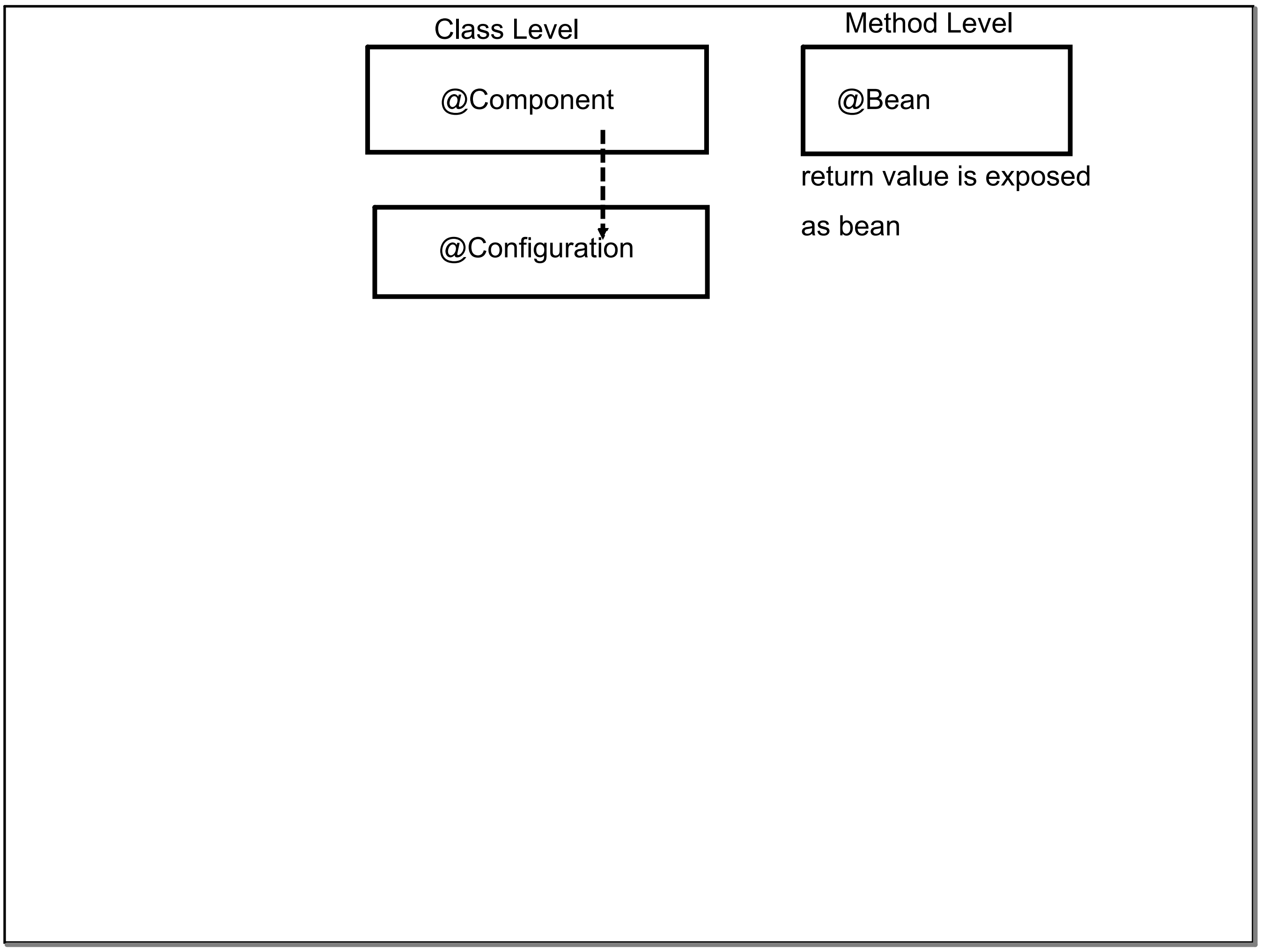
Literal Value : @Value

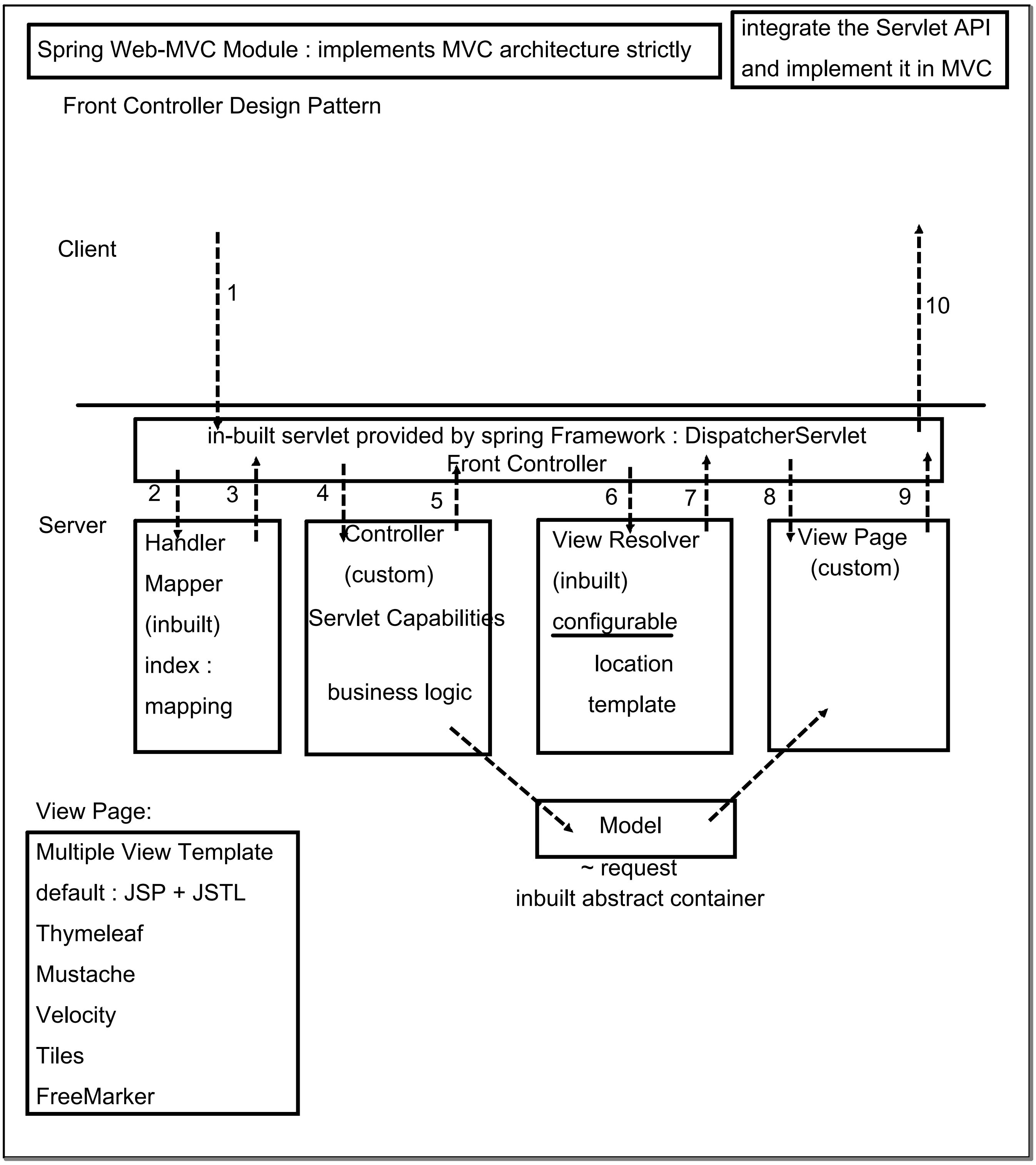
@Scope : for defining the scope of the bean

Pure Java Config

XML file will be replace with a Java class




Class Level
@Component
@Configuration
Method Level
@Bean
return value is exposed
as bean


Spring Web-MVC Module : implements MVC architecture strictly
integrate the Servlet API and implement it in MVC
Front Controller Design Pattern
Client
1
10
in-built servlet provided by spring Framework : DispatcherServlet
Front Controller
Server
2
3
4
5
6
7
8
9
Handler Mapper (inbuilt) index : mapping
Controller (custom) Servlet Capabilities business logic
View Resolver (inbuilt) configurable location template
View Page (custom)
Model
~ request
inbuilt abstract container
View Page:
Multiple View Template
default : JSP + JSTL
Thymeleaf
Mustache
Velocity
Tiles
FreeMarker

Dynamic Web Project (Servlet-API)

+

Spring Framework(jar file) + JSTL

C : Servlets

V : JSP

M : data structure

C : POJO (Servlet capabilities)

Configuration Servlet-API spec

1. Registration of Servlet (DS)
2. Map DS with all URL

web.xml

Need to map/register the spring config with DS

naming convention

Spring part

1. Config for Bean Factory
2. Config for View Resolver

<servlet-name>-servlet.xml

View Resolver : Resolve the details view page

eg : Controller returns only page name

"index" (string)

prefix : location

suffix : View Template (ext)

```
<property name="prefix" value="/WEB-INF/views/"/>
<property name="suffix" value=".jsp"/>
```

/WEB-INF/views/index.jsp

Controller : POJO, registered with HAndler Mapper

$$\begin{tikzcd}\text{@Component} \arrow[d] \\ \text{@Controller}\end{tikzcd}$$

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

$$\text{dispatcher-servlet.xml} \dashrightarrow \text{Java class}$$

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

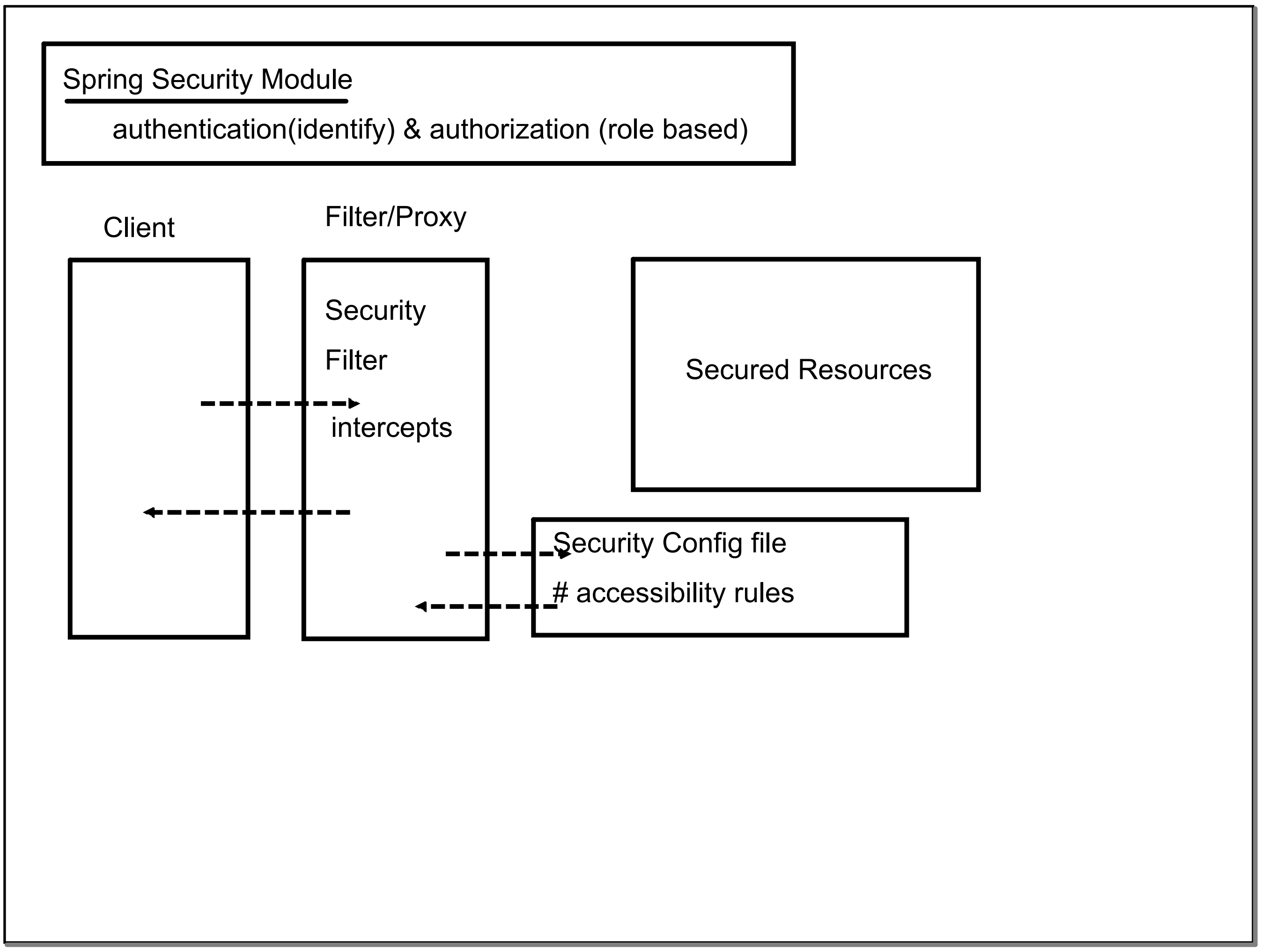
javax : preferrable : prevent Vendor locking

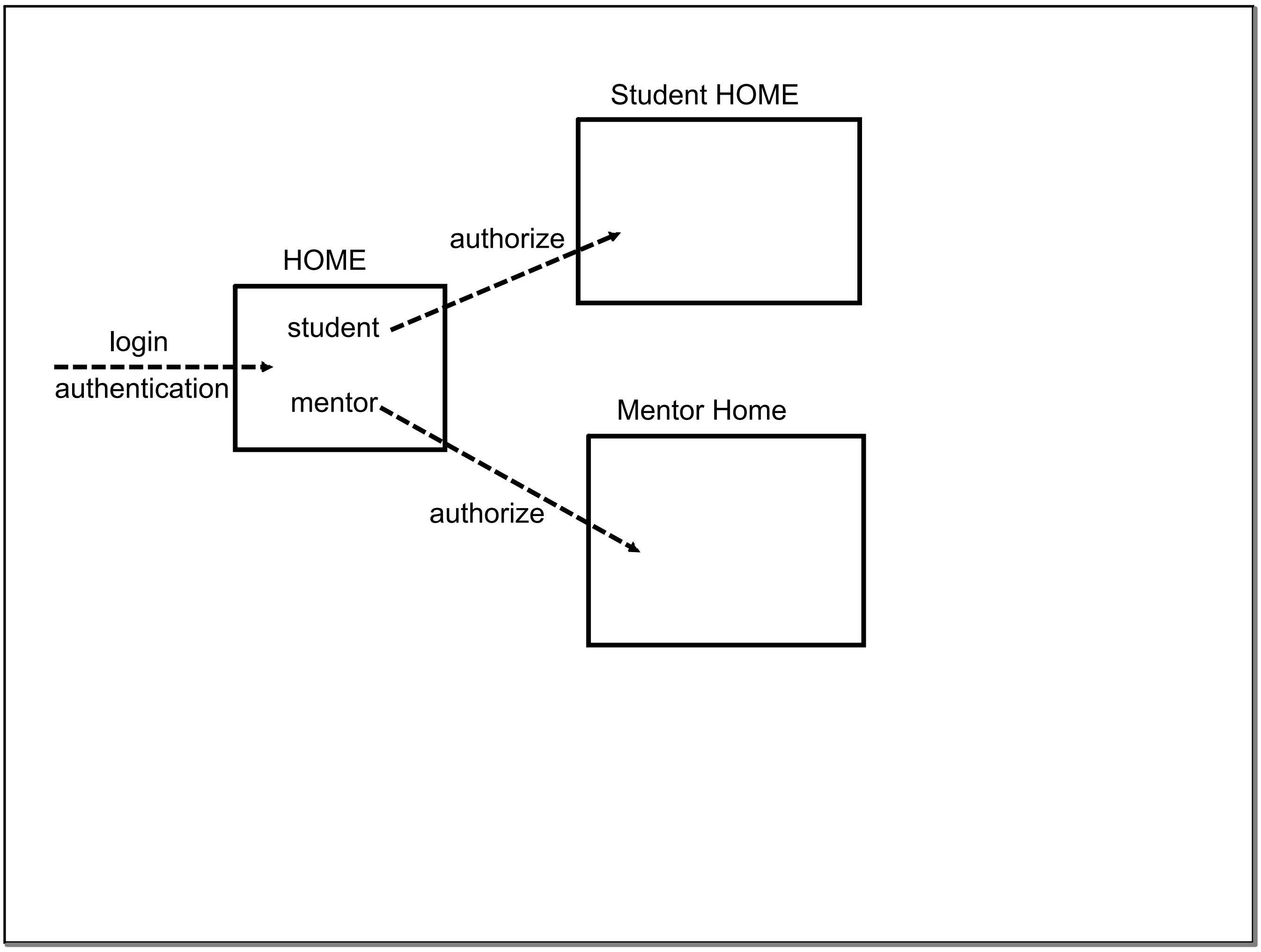
Validation

Client-Side Validation : HTML5 + JS

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

Custom Validation Annotation


Spring Security Module
authentication(identify) & authorization (role based)
Client
Filter/Proxy
Security
Filter
intercepts
Secured Resources
Security Config file
# accessibility rules


Student HOME
authorize
HOME
student
login
authentication
mentor
Mentor Home
authorize

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 out 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

inbuilt validation method is provided

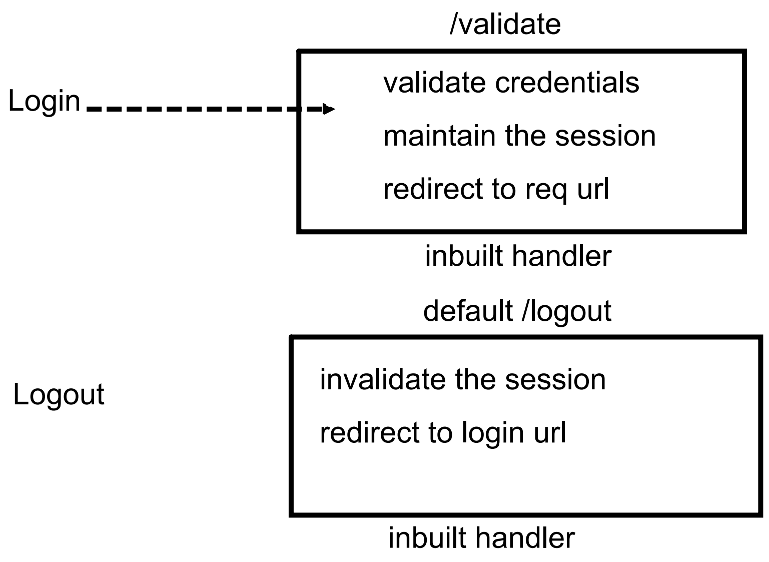
# username

# password

# submission method : POST

logout option

spring form : submit button



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 relevent 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

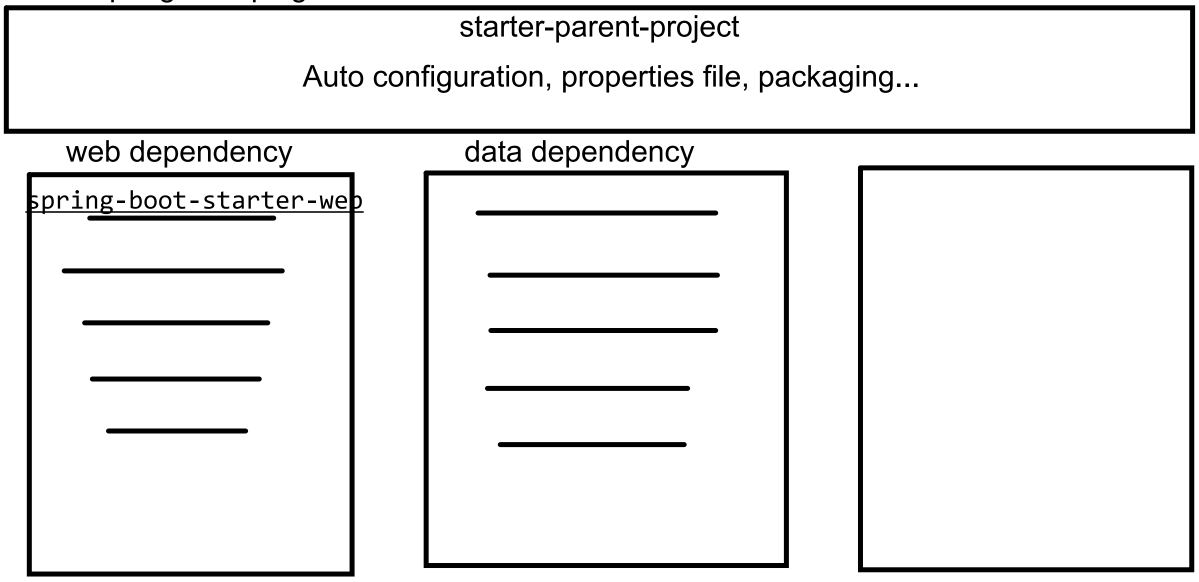
# maven : does not need any support of maven on developer machine

# 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

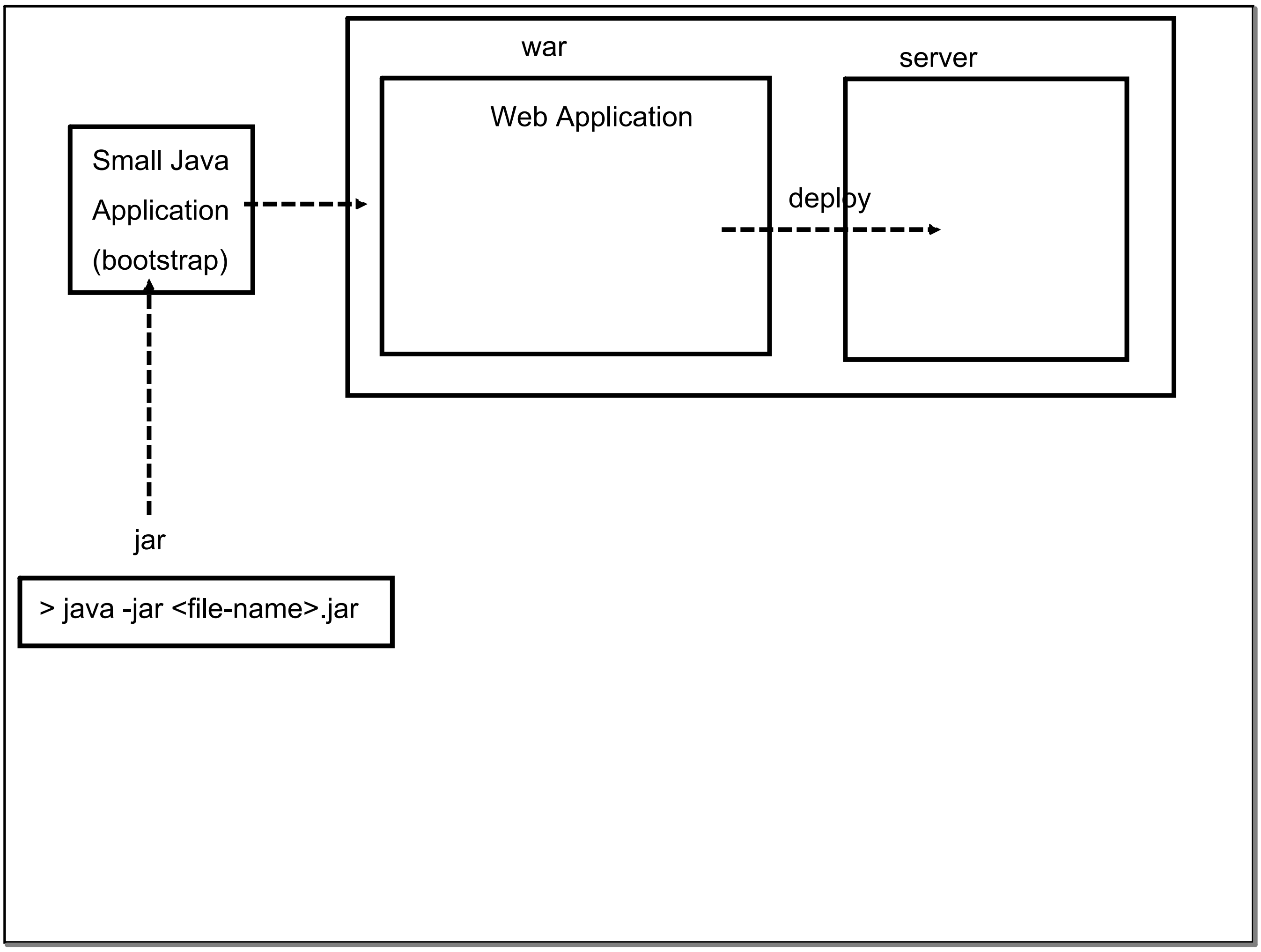


webapp : jsp+jstl view template

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

templates : home for view page

application.properties : key-value (custom config)


war
server
Web Application
Small Java
Application
(bootstrap)
deploy
jar
> java -jar <file-name>.jar

# 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

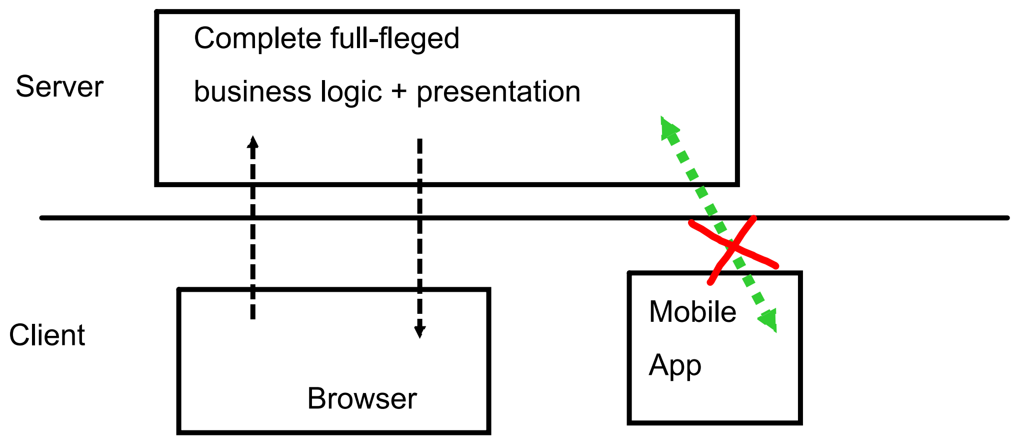
mvnw <option> : if maven is not installed

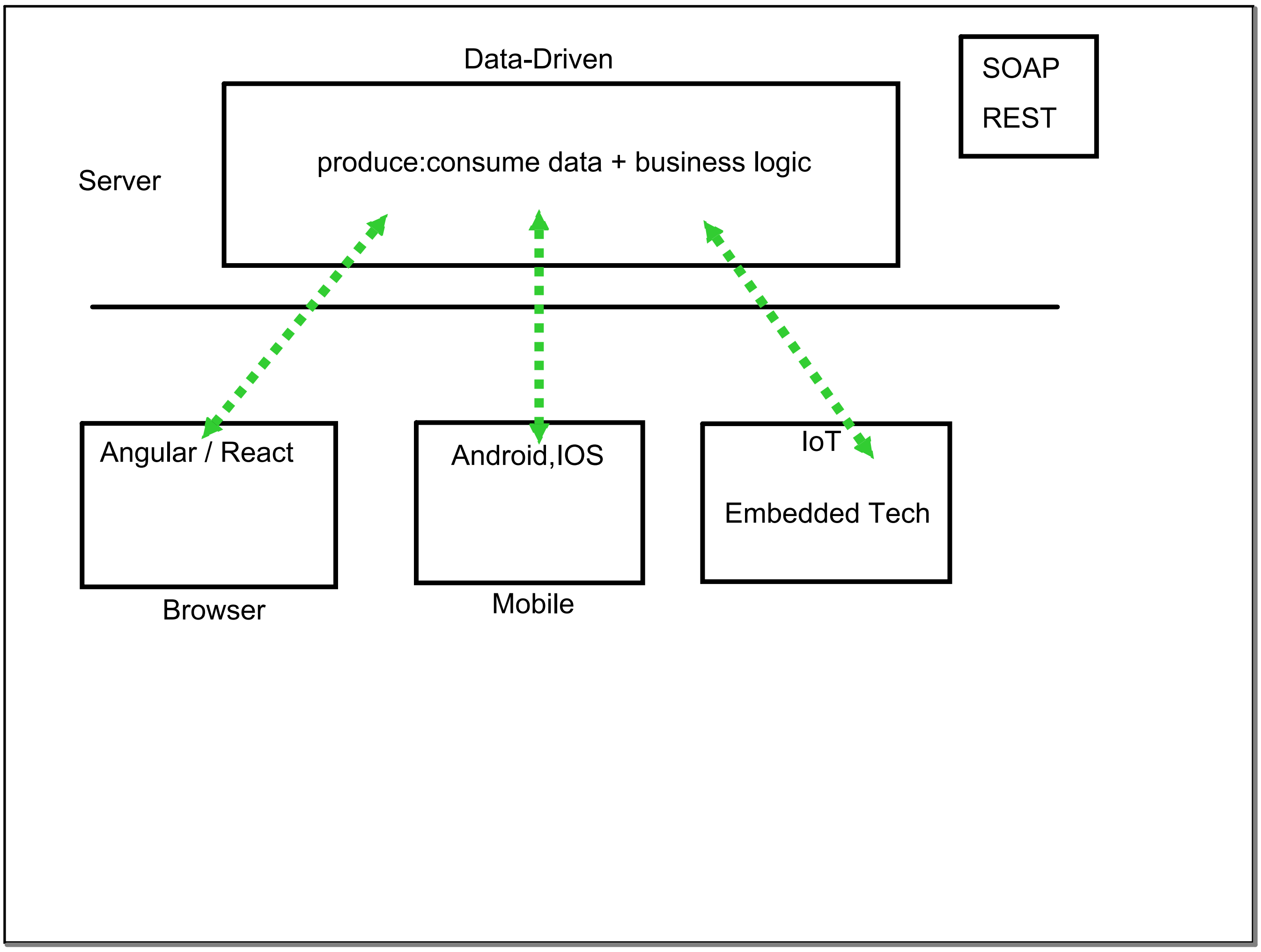
Converting Spring application to Spring boot

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

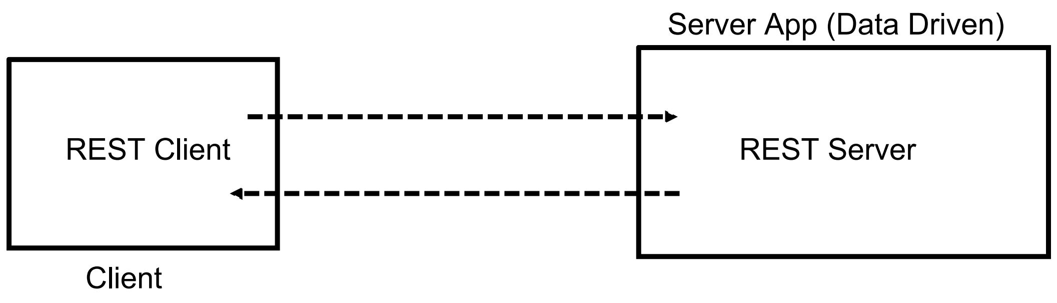
RESTful : Data-Driven Application




Data-Driven
SOAP
REST
Server
produce:consume data + business logic
Angular / React
Browser
Android,IOS
Mobile
IoT
Embedded Tech

SOAP : Legacy :

Functional (API) : Object



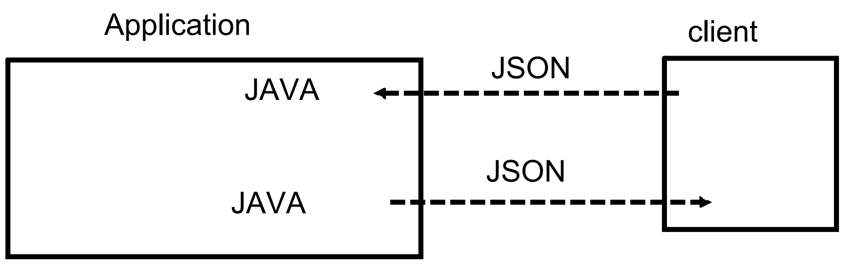
1. How do i talk with Server App?
2. What is the data format ?

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

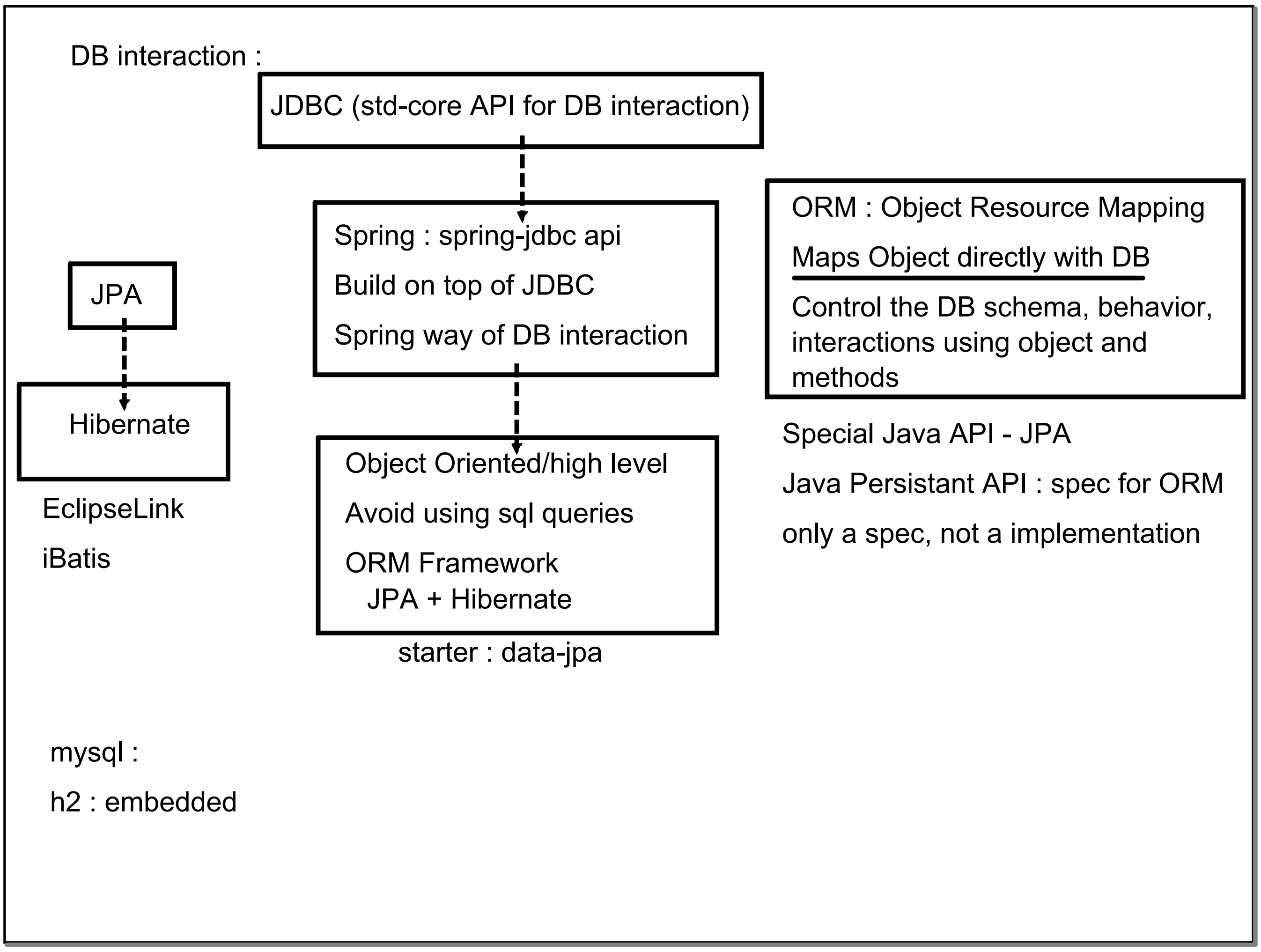
Create controllers that can work on REST Protocol

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



DB interaction :

JDBC (std-core API for DB interaction)

Spring : spring-jdbc api

Build on top of JDBC

Spring way of DB interaction

Object Oriented/high level

Avoid using sql queries

ORM Framework
JPA + Hibernate

starter : data-jpa

JPA

Hibernate

EclipseLink

iBatis

ORM : Object Resource Mapping

Maps Object directly with DB

Control the DB schema, behavior, interactions using object and methods

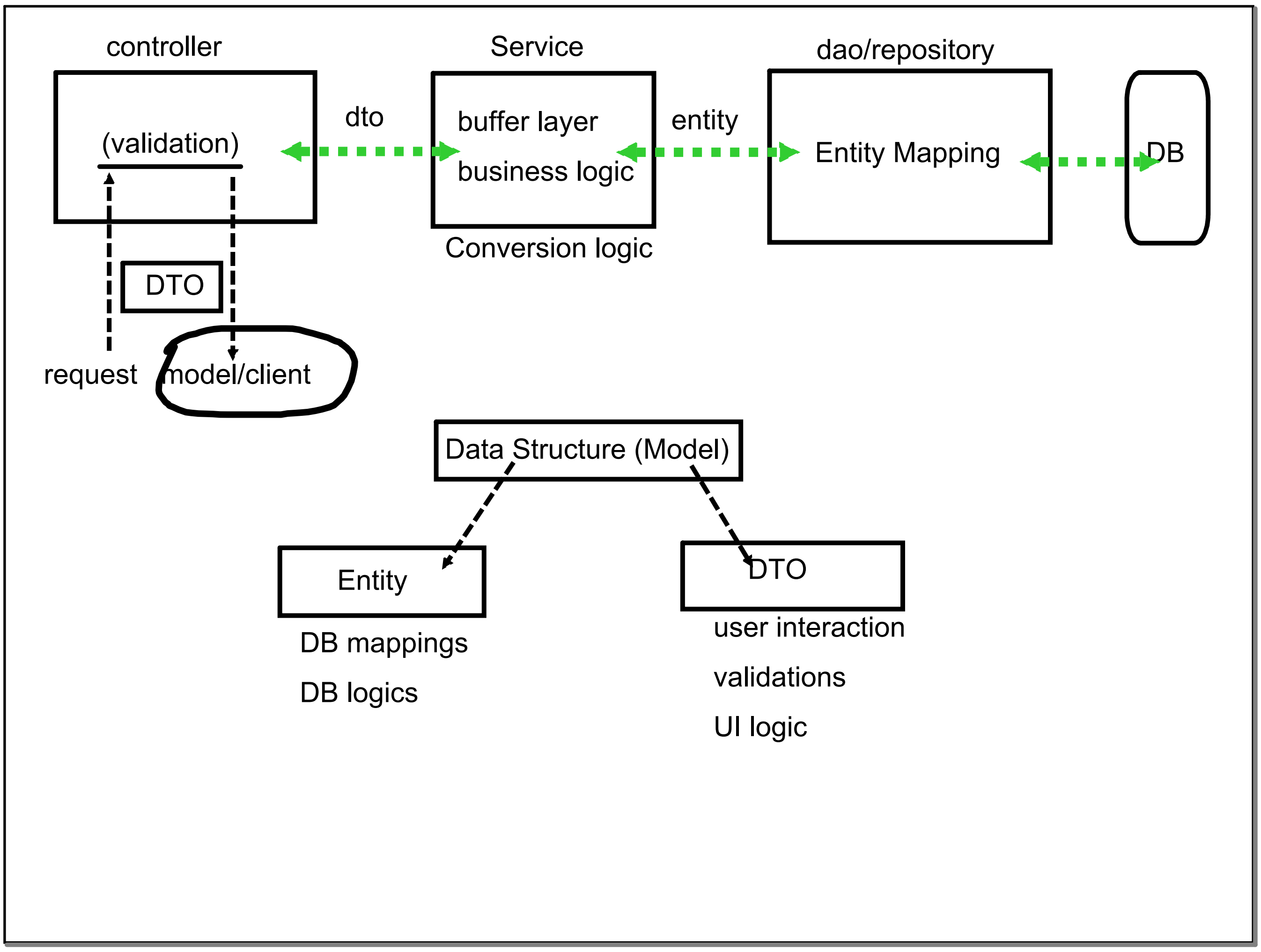
Special Java API - JPA

Java Persistant API : spec for ORM

only a spec, not a implementation

mysql :

h2 : embedded


controller
(validation)
dto
Service
buffer layer
business logic
Conversion logic
entity
dao/repository
Entity Mapping
DB
DTO
request
model/client
Data Structure (Model)
Entity
DB mappings
DB logics
DTO
user interaction
validations
UI logic

Employee info

name

email

contact

salary :

basic pay

hra

ta

da

pfdeduction

netsalary

Base Class

name

email

contact

Input : name, email, contact, basic-pay DTO

Save: id,name, email, contact, basic-pay, hra, ta, da, pfdeduction entity

Output : id, name, email, contact, netsalary DTO

Best Rest Practices:

add : /addEmployee : /employees : POST

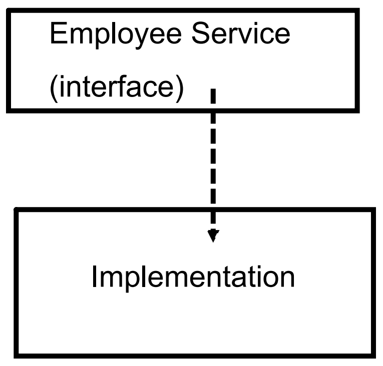
delete /deleteEmployee : /employees/{id} : DELETE

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

fetch all : /employees : GET

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

<plural form of entity> : change the HTTP verbs

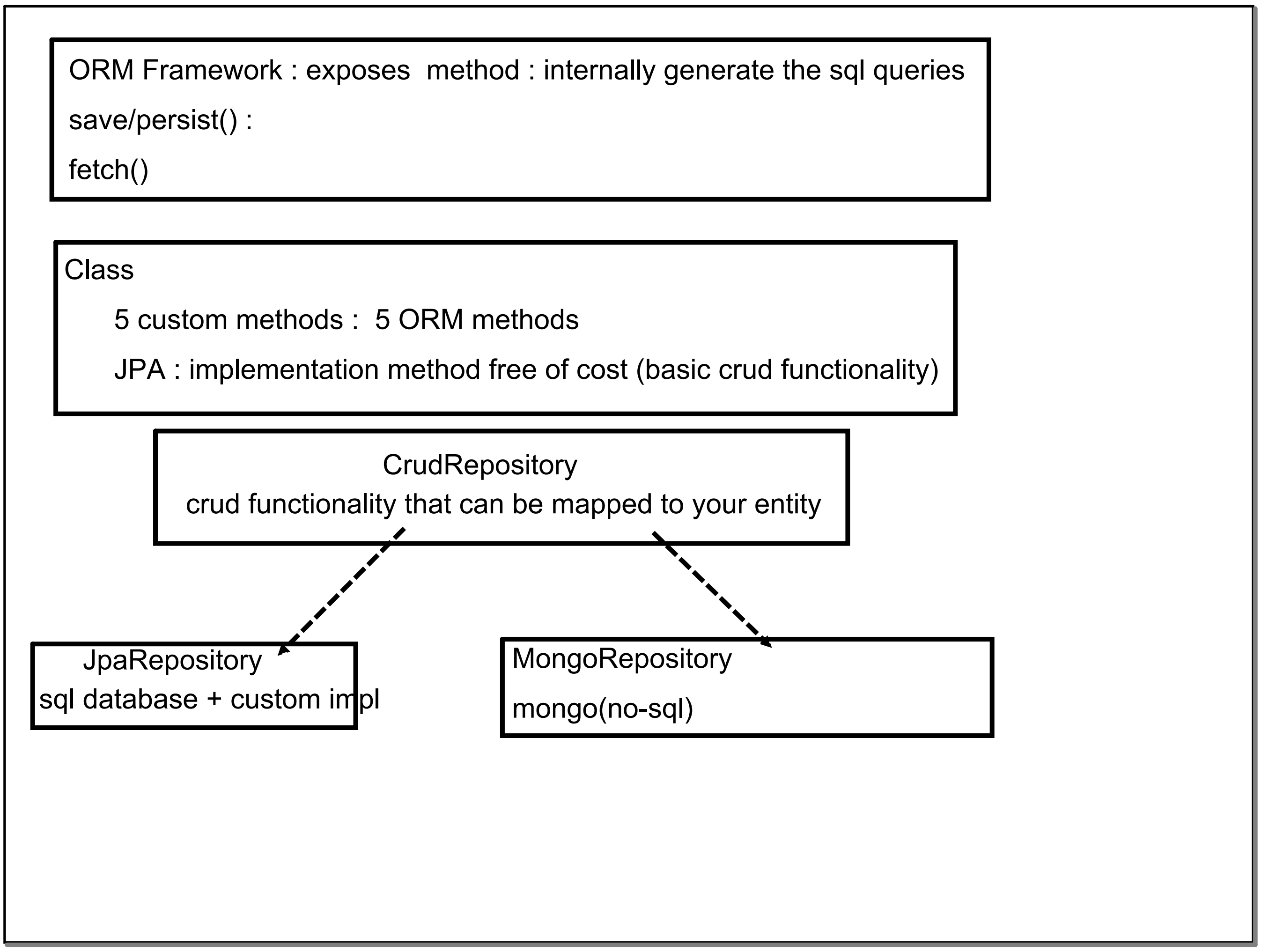


Complete DB Schema will be defined as Java Classes

Class (Entity) : Table

data - member : colomn

ORM framework, mapping is required


ORM Framework : exposes method : internally generate the sql queries
save/persist() :
fetch()
Class
5 custom methods : 5 ORM methods
JPA : implementation method free of cost (basic crud functionality)
CrudRepository
crud functionality that can be mapped to your entity
JpaRepository
sql database + custom impl
MongoRepository
mongo(no-sql)

Create custom interface, inherit the JpaRepository interface

# Employee (Entity)

# Primary Key type

select * from employee

select * from student

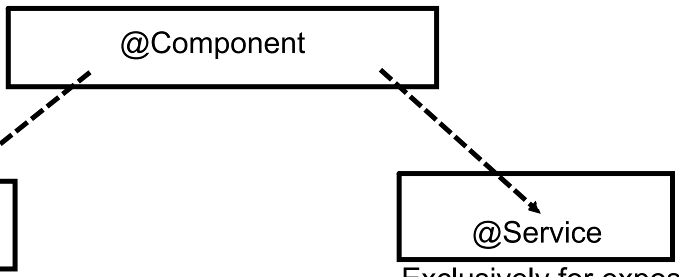
select * from books

ORM method

generate a sql query internally :

mysql,oracle, postgre : variation in dialect

POSTMAN : REST Client Application



Exclusively to expose
bean of dao/repository
# Converts the jdbc/sql
exception into jpa excp.

Exclusively for expose
beans of service
Nothing addn right now

Handling the exception : spring-way

#From client perspective exception should be handled in Controller

# Delegated method for handling exceptions

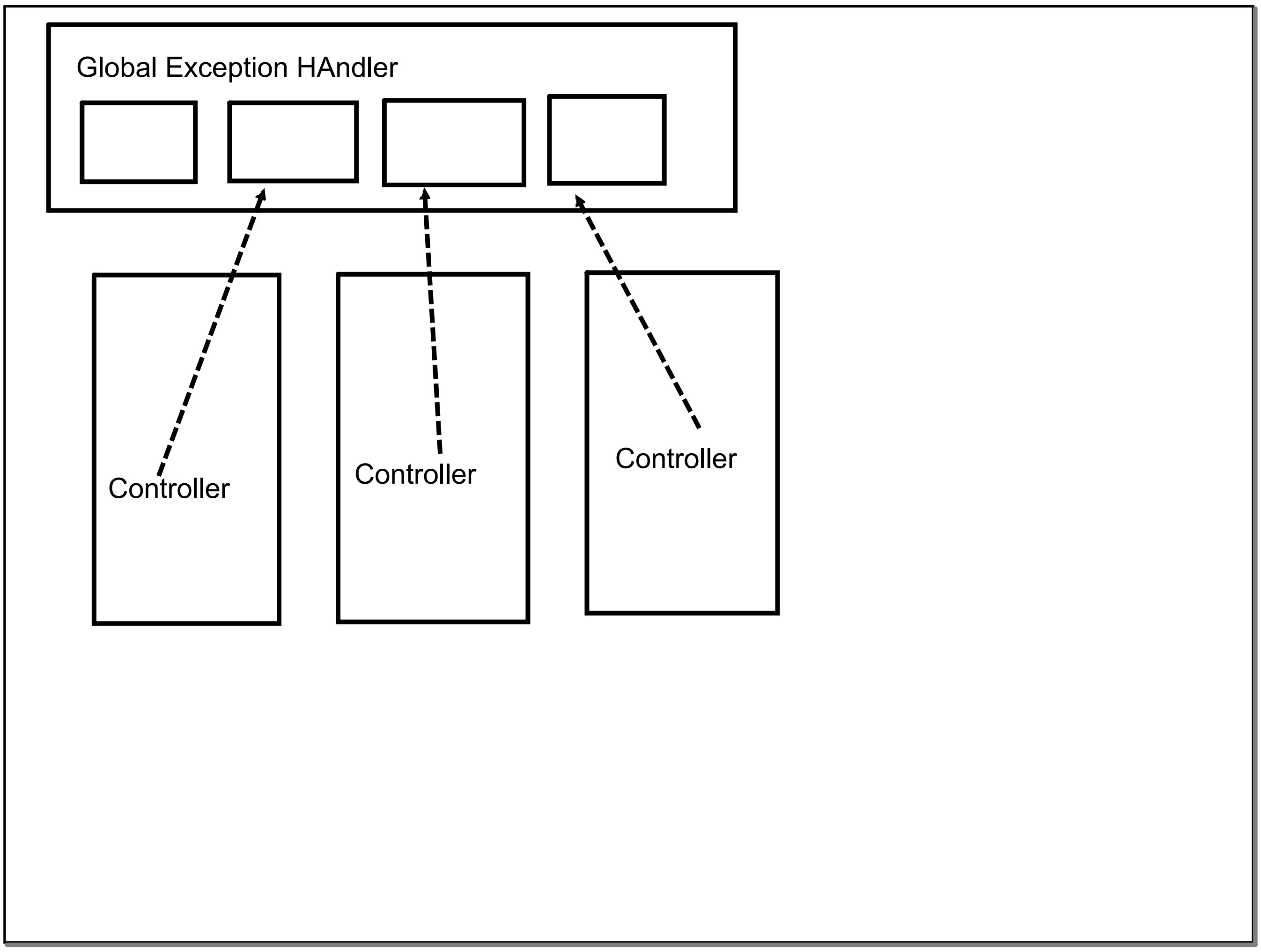
# We must have a specialized DTO for responding in case of exception

To respond with appropriate HTTP Status Code:

Wrapping the response in a object ResponseEntity

HTTPStatus code can be specified explicitly

#Don't return the raw data


Global Exception HAndler
Controller
Controller
Controller

JpaRepository : Basic Crud functionality is exposed

custom, specialized DB requirement