Spring Framework:

Framework for building Java Applications

Simpler and lightweight alt to J2EE

Early version of J2EE complex object management

Multiple deployment descriptors

Multiple interface

Poor Performance

Rod Johnson

Lightweight Object management tool : Object Factory/Bean Factory/Application Context

Spring :

=> highly Modular

=> Independent to be used : loose coupling among modules

=> Java POJOs : Lightweight dev process

Relationship among resources of Spring application

1. IoC : Inversion of control
2. DI : Dependency Injection
3. AOP : Aspect Oriented Programming (proxy)

Modules

- Core Container : Bean Factory
- Web Layer : MVC Framework
- Data Access Module : JDBC/ORM/Transaction
- Infra : AOP/Messaging
- Testing : JUnit/Mocking

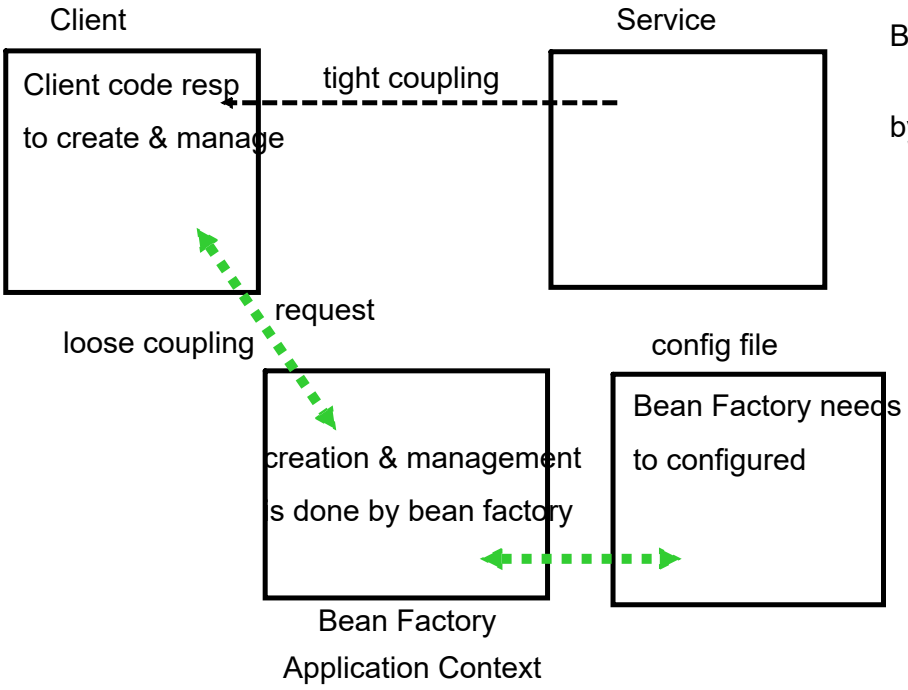
Spring Projects:

- Spring Data
- Spring Cloud
- Spring security
- WebFlux

1. IoC : Inversion of Control

Outsourcing the Creation and Management of Object :

Object Factory/Bean Factory



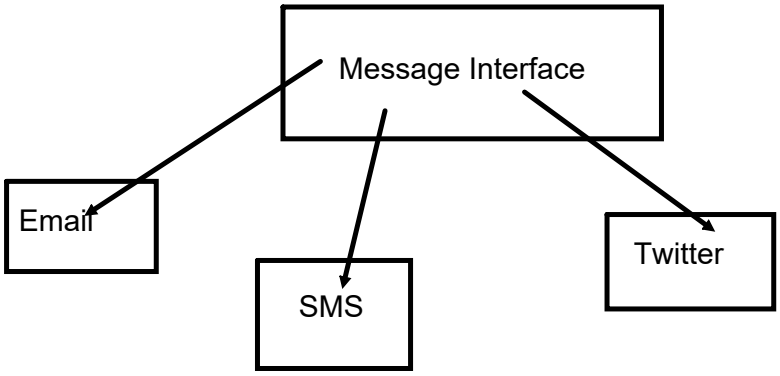
Beans :

Objects managed by spring containers

Spring Jar files : https://repo.spring.io/release/org/springframework/spring/

Configuration of Bean Factory

1. XML based config (legacy)
2. Annotations
3. Pure Java Code



```
<bean id="mservice" class="com.wf.training.spring.service.EmailService">
```

key: by which bean is exposed

class: whose bean we want to create

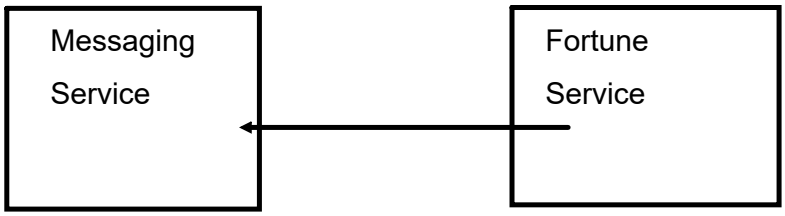
DI : Dependency Injection

Creation of complex object might have dependency on other object :

injecting those dependency with the help of Spring Container

XML based config:

1. Constructor based
2. Setter based



literal values should not be integrated in config file

==> property file to keep the literal values

KEY - VALUE PAIRS

Referred by SpEL

Scope of Beans

Default scope : Singleton

Only one instance which shared among all calls

Scope possibility :

singleton

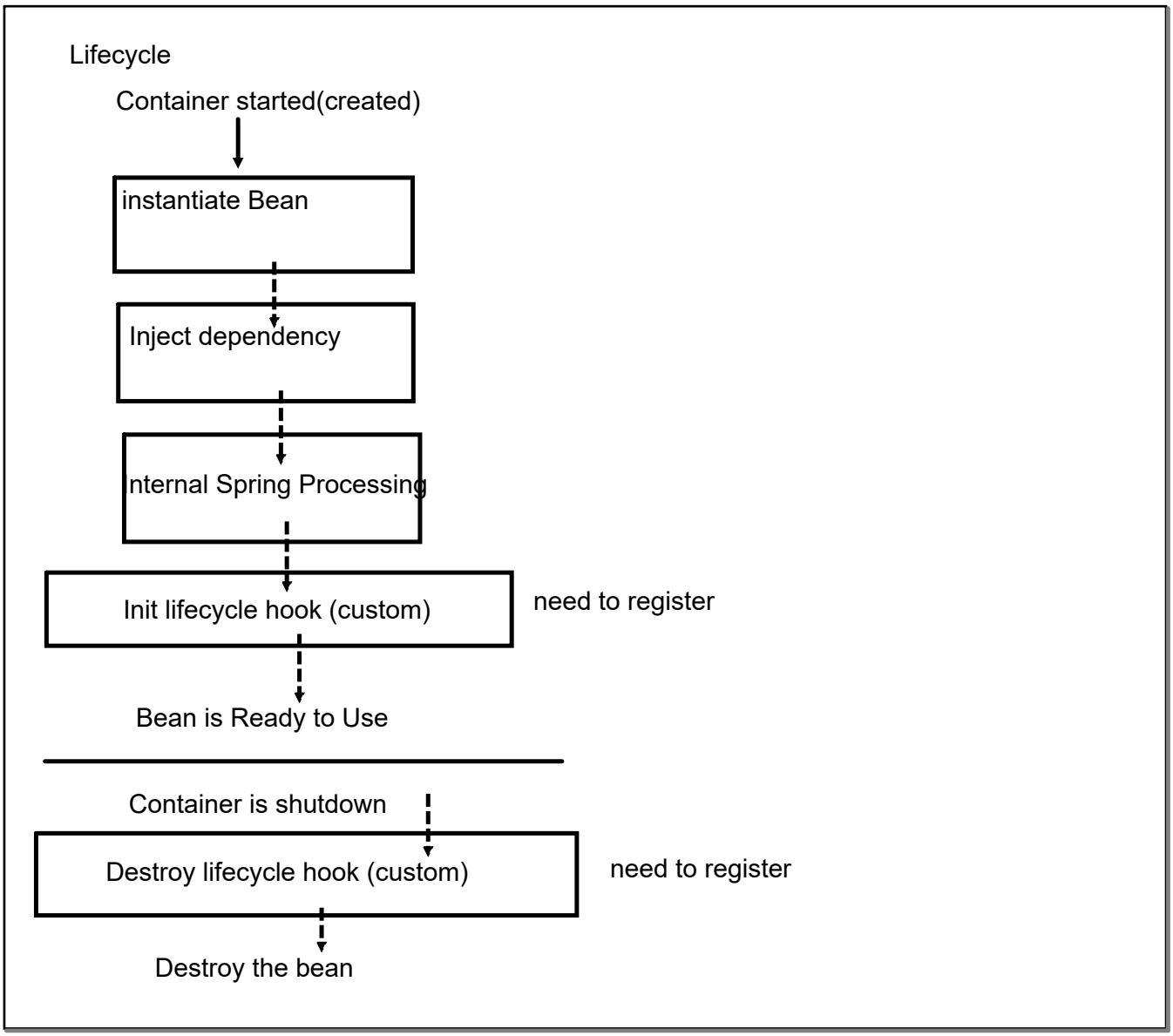
prototype : a new bean would be created

Web Context :

request

session

global-session

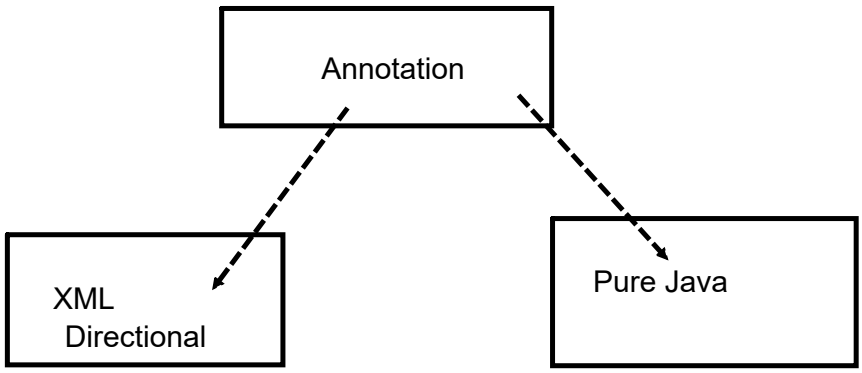

Lifecycle
Container started(created)
instantiate Bean
Inject dependency
nternal Spring Processing
Init lifecycle hook (custom)
need to register
Bean is Ready to Use
Container is shutdown
Destroy lifecycle hook (custom)
need to register
Destroy the bean

Life cycle hook method

1. any name
2. any access modifier
3. not static
4. they may return values but can't capture
5. No parameter

Prototype : Spring container does not maintain lifecycle

Annotation based config



@Component : informing spring to create and manage bean of that class

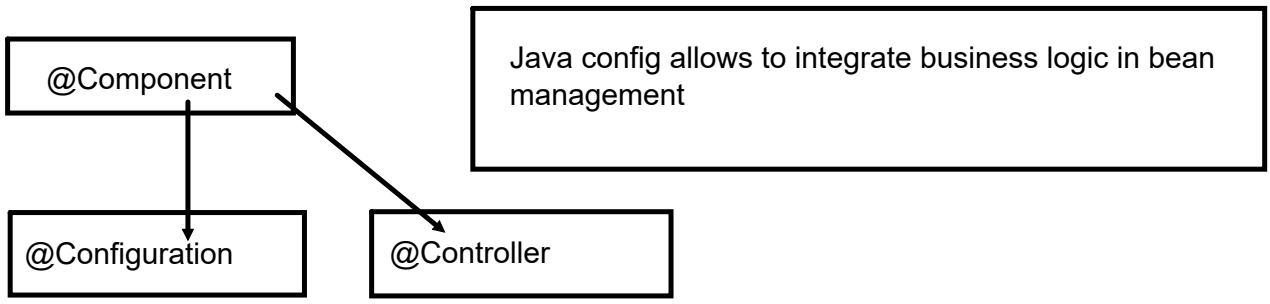
Every bean must be exposed by an id (Class name (with first char small) is default id)

Annotation based DI

1. Constructor
2. Setter
3. Field

Java Based Configuration

XML file is replaced by a Java Config



Developing Web Based Application using Spring Framework

spring web-mvc

spring-core

Spring also uses Servlet-API (Servlet Spec)

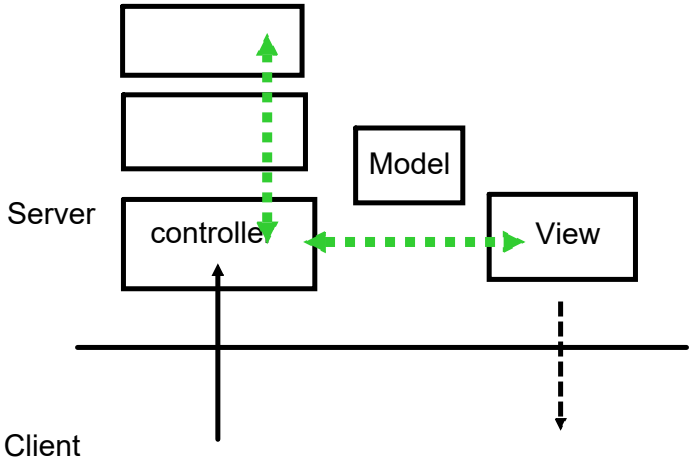
spring container

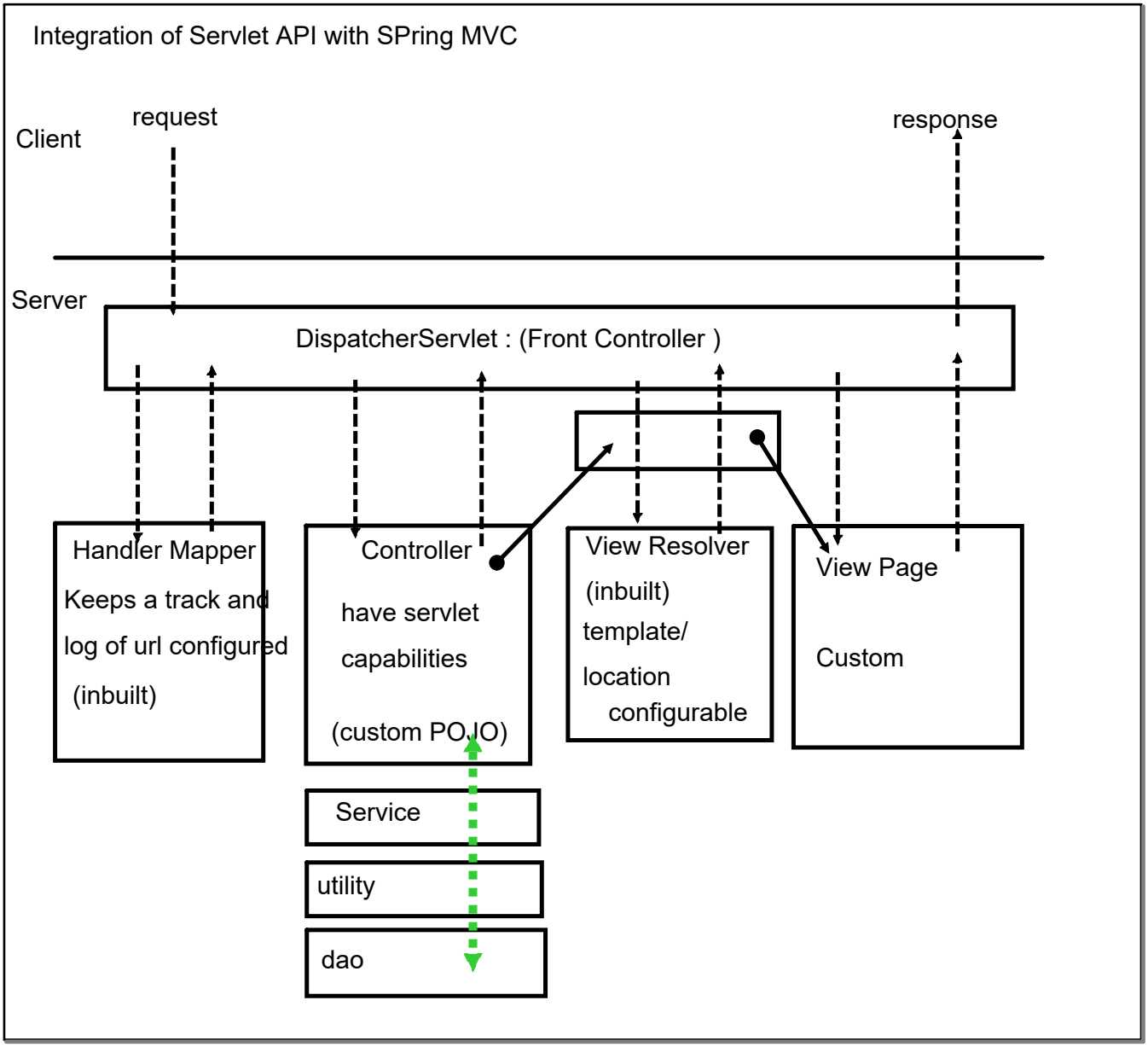
highly abstraction : POJOs

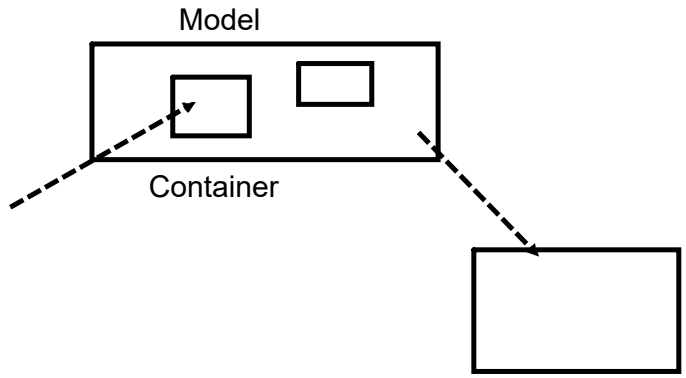
Spring Web : MVC Design Pattern

Servlet API :

Strictly exposes the MVC : standardized, uniform




Integration of Servlet API with SPring MVC
Client
request
response
Server
DispatcherServlet : (Front Controller)
Handler Mapper
Keeps a track and
log of url configured
(inbuilt)
Controller
have servlet
capabilities
(custom POJO)
View Resolver
(inbuilt)
template/
location
configurable
View Page
Custom
Service
utility
dao



Spring MVC:

Multiple View Templates :

Traditional Default : JSP + JSTL (lib added for jsp+jstl)

- Thymeleaf
- FreeMarker
- Tiles
- Velocity
- Mustache

Two configurations

1. Servlet API based : Need register and configure the Dispatcher Servlet
2. Spring config for that servlet, helpers for servlet
   xml file needs to tightly binded with servlet
   naming convention : <servlet-name>-servlet.xml

eg:

dispatcher-servlet.xml

View Resolver : Creating and exposing a bean of View Resolver

Controller revert back : Name of view page

eg: "my-view"

/WEB-INF/views/my-view.jsp

URL Mapping must be unique across the application

Class level URL Mapping can be done

Multiple Urls mapped to same method

Default mapping

Fallback

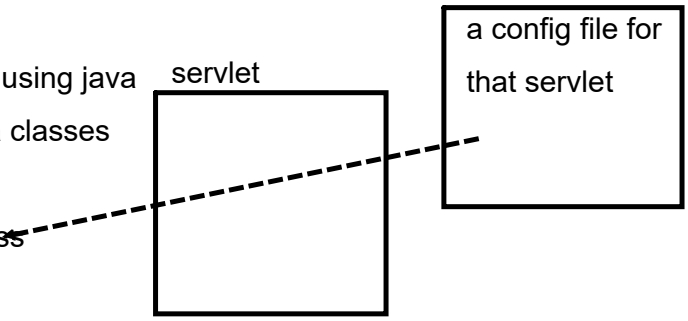
@RequestMapping() maps all http verbs by default (GET/POST/PUT/DELETE)

Spring Web MVC project configured using java

XML files would be replaced by Java classes

web.xml : Java class

dispatcher-servlet.xml : Java class



web.xml ~

Register the DS : inherit a spring framework class

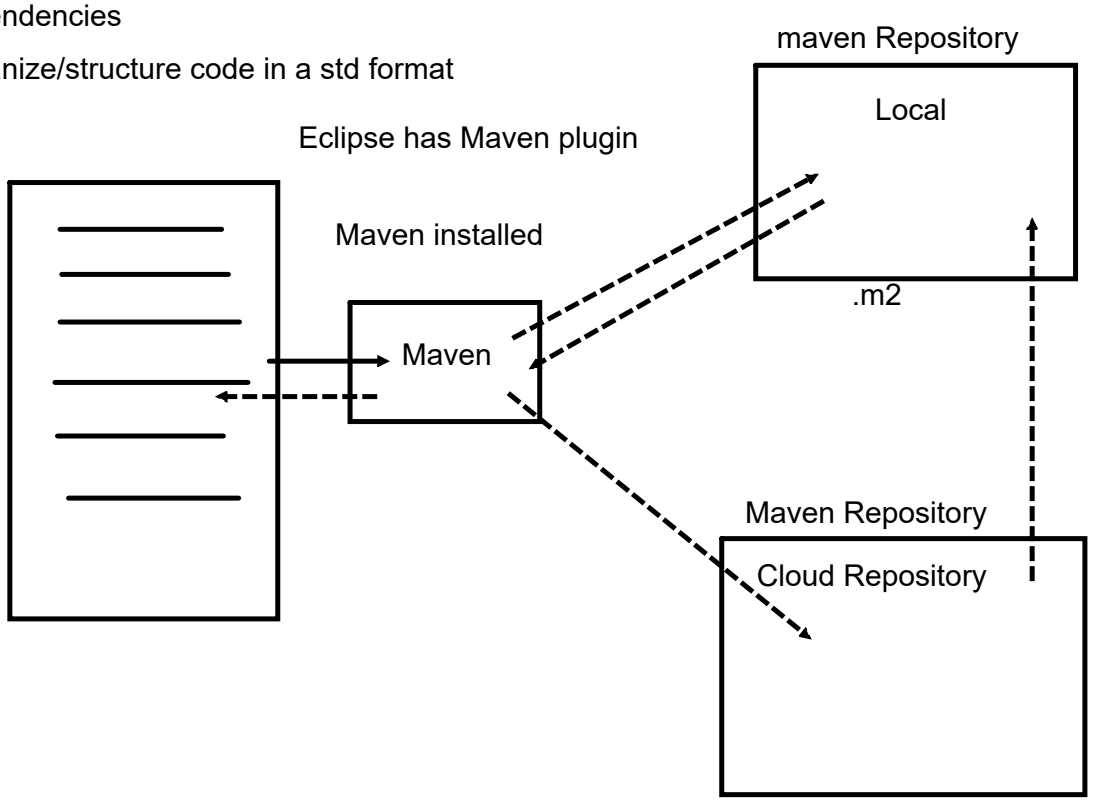
Other config to be done by overriding the abstract method

Traditional way : jar file management

Tools Project Management Tools

Ant, Gradle, Maven

1. dependencies
2. Organize/structure code in a std format



Each Maven Project is identified by GAV coordinates

GROUP : organisation (reverse domain)

Artifact ID : project name

Version : version of project

Dependency:

1. spring webmvc
2. servlets
3. jsp

plugin :

maven war plugin

maven webapp archetype project
changed java 1.5 to 1.8
added the Server Runtime Support
Added maven dependency
deleted the web.xml
add maven war plugin

Forms in Spring

Spring provides library that exposes tags for form handling

Spring Form tags :

It can be mapped with Model Object

Inherent Security : CSRF attack

Validation

Client Side : Javascript

Server Side :

Java Validation API: Validate the java object

Just a spec (interfaces) but no implementation is there

Most Popular implementation : Hibernate validators (not ORM)

Native JAva Validation API :

Implementation Annotation not recommended : prevent vendor locking

Added dependency hibernate validator
Decorated the model flds with validation annotation
Need to tell spring to validate and get the result
need to check the result and decide
show the message

Model classes shall not use
the primitive type
wrapper types

data ---------> String

1. eg _____First_____

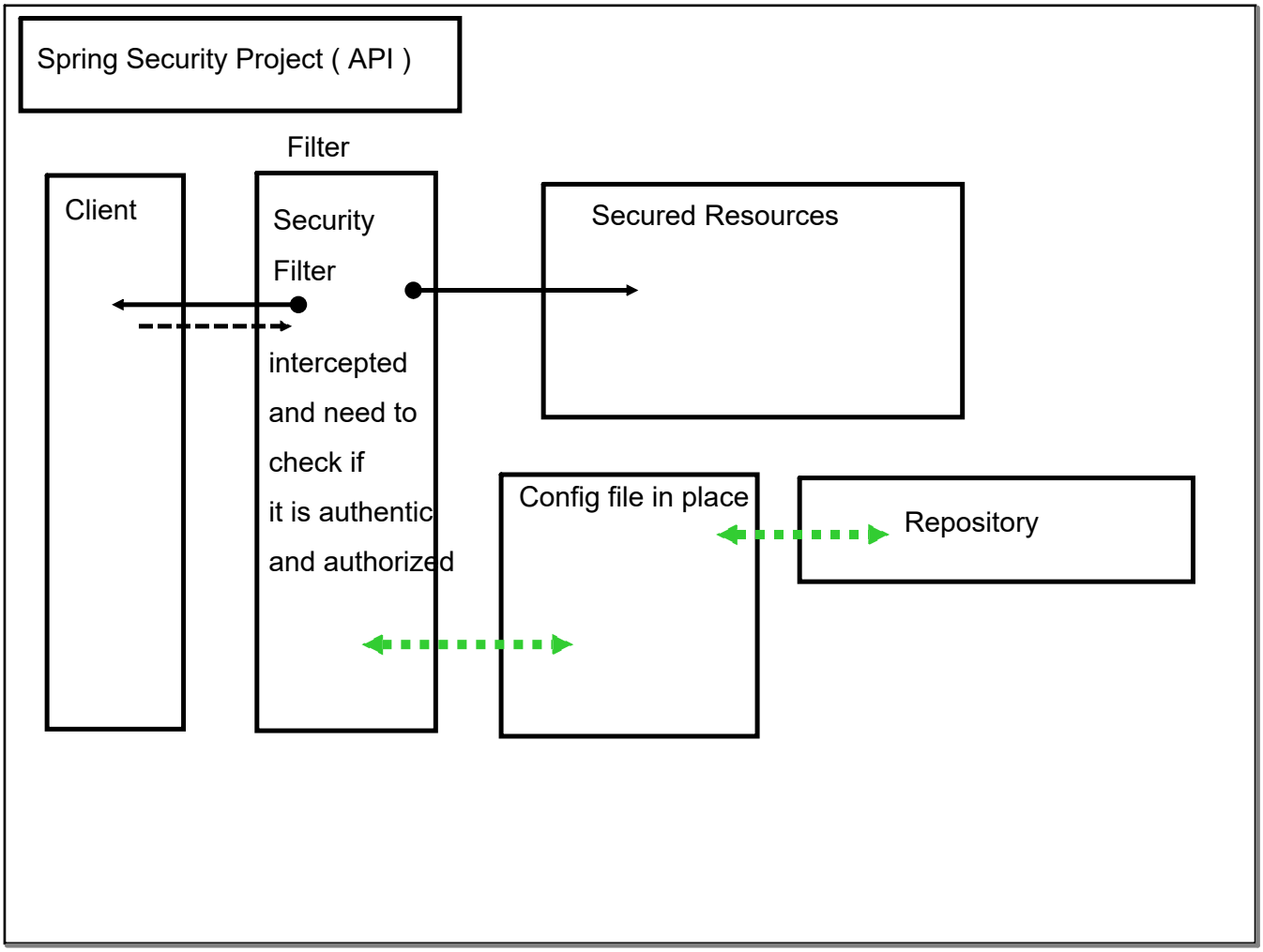
2. Empty string : NULL

proxy method

Custom Validation Annotation

Create an annotation

interface

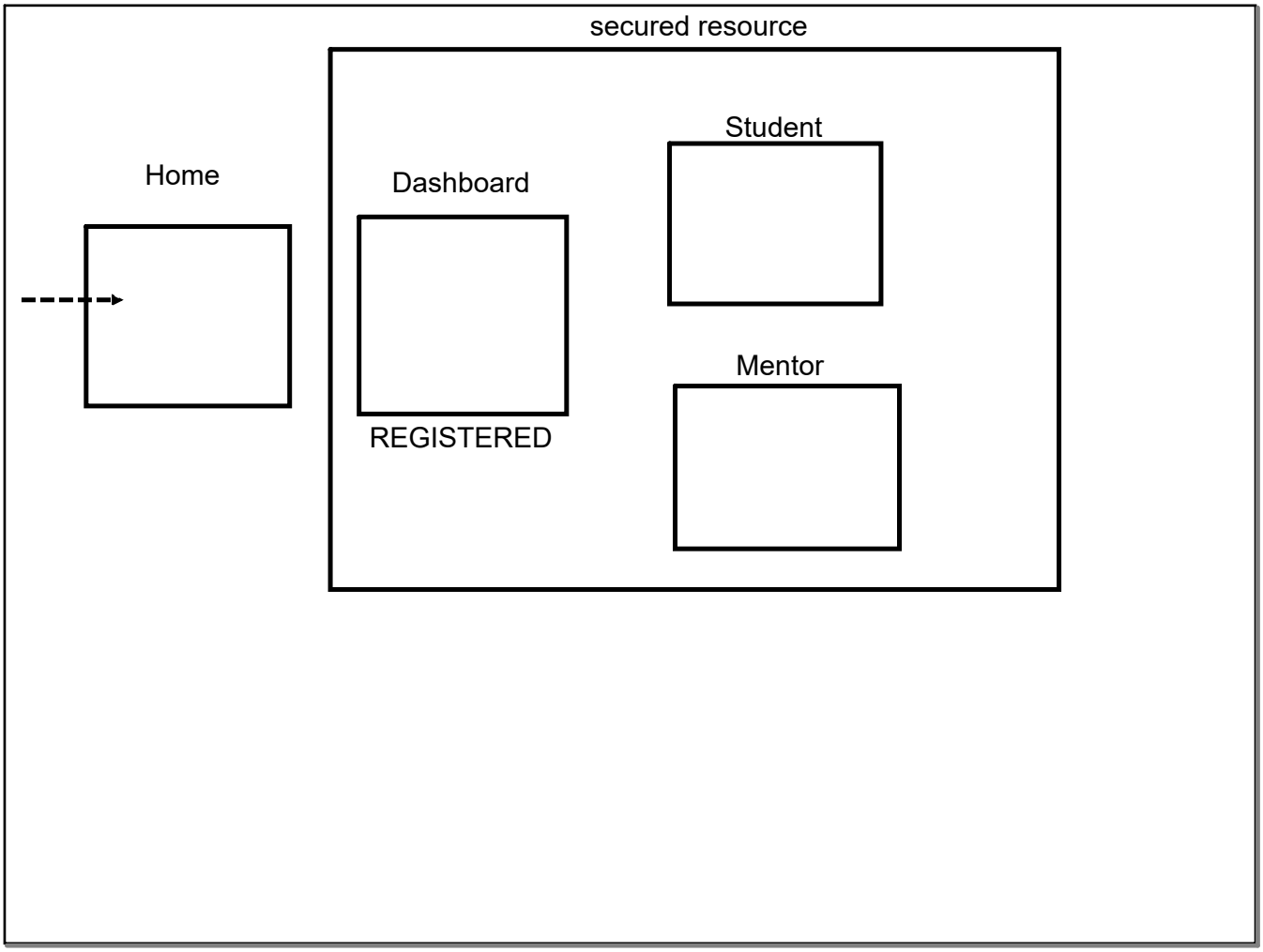

Spring Security Project (API)
Filter
Client
Security
Filter
Secured Resources
intercepted
and need to
check if
it is authentic
and authorized
Config file in place
Repository

Confirmation of all these are through config file

1. if request is not of secured resource, direct access

2. is for secured resource : filter exposes a login form (inbuilt login form provided by spring security API)

3. Credential submission is confirmed by config file, reverted back to filter


secured resource
Home
Dashboard
REGISTERED
Student
Mentor

Dependency :

spring-security-web (filter)

spring-security-config (add config)

spring-security-taglibs

1. Initialize the security Filter

Authentication an take place in three different ways

1. httpbasic : instruct the browser to generate a login dialog-box (not-recommended)
2. formlogin : (default) default login form / customization
3. basicauth: Token based (Data-driven) REST APIs (Stateless)

in case of bad credentials /custom-login?error