Spring Training

CREST
DATA SYSTEMS

Day - 1

Introduction to Spring framework & Spring Modules

CREST DATA SYSTEMS

What is spring framework

- The most popular application development for java.
- Simpler and lightweight, easier to develop than JakartaEE
- Provides a large number of helper classes to make things easier

What is in the latest version of the spring

- Latest version is spring 5
- Updated minimum requirement to Java 8.0 or higher
- Deprecated legacy integration for Tiles, Velocity, portlets, Guava etc.
- Upgraded Spring MVC to new version of Servlet API 4.0
- Added new reactive programming framework: Spring WebFlux

CREST DATA SYSTEMS

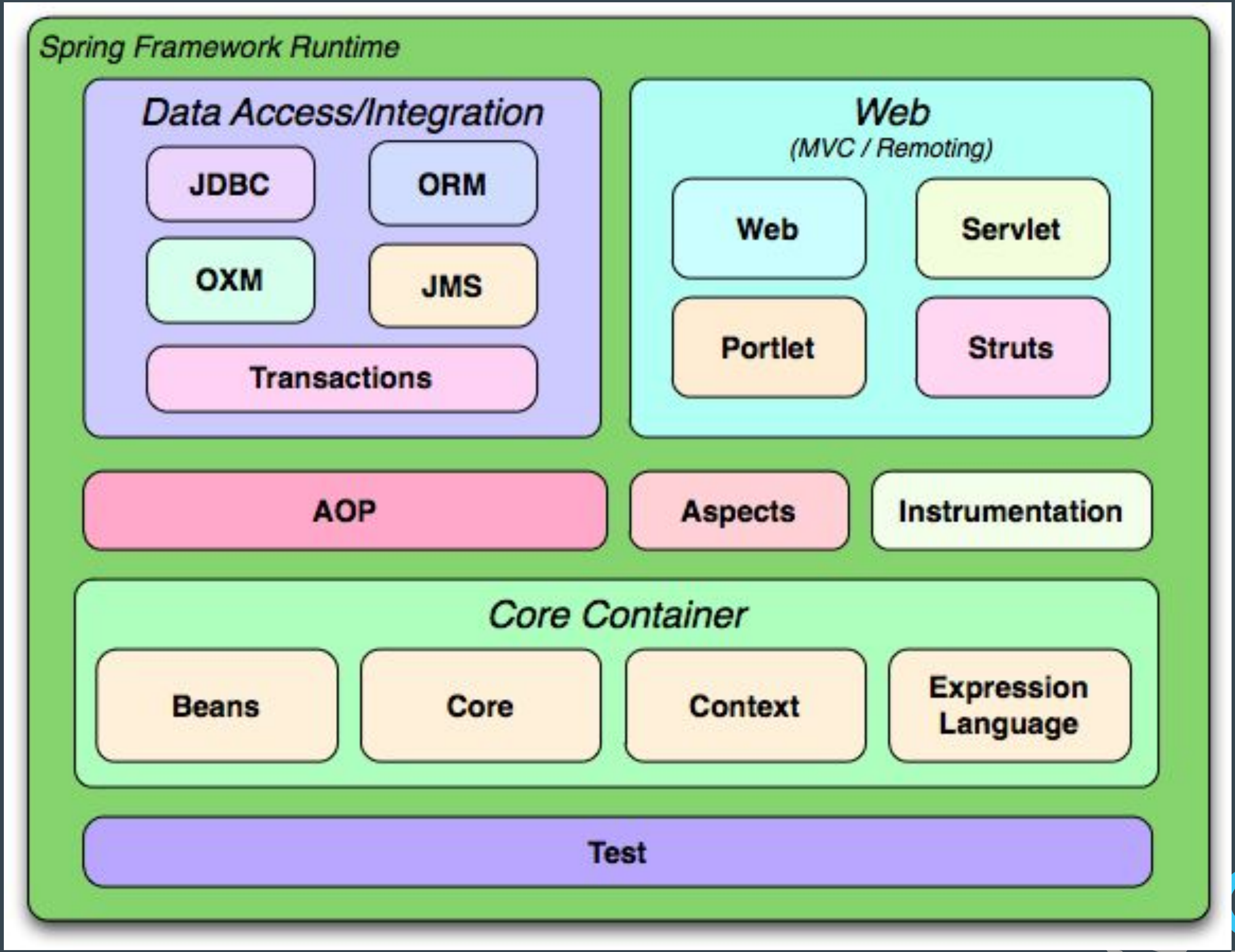
Goals of Spring

- Lightweight development with JAVA POJOs (Plain Old Java Objects)
- Dependency injection to promote loose coupling
- Declarative programming with Aspect-Oriented-Programing (AOP)
- Minimize boilerplate java code
- Promote good software development practice.

Spring modularity

- Since its foundation, the framework has had a particular focus on modularity.
- It is an important framework characteristic because it makes the framework an excellent option for different architectural styles and different parts of applications.
- We can use the framework as we need and integrate it with a wide range of specification and third-party libraries.
- It is divided into various modules. The main modules are as follows:
  1. Spring Core
  2. Spring Data
  3. Spring Security
  4. Spring Cloud
  5. Spring Web-MVC

CREST DATA SYSTEMS


Spring Framework Runtime
Data Access/Integration
JDBC
ORM
OXM
JMS
Transactions
Web
(MVC / Remoting)
Web
Servlet
Portlet
Struts
AOP
Aspects
Instrumentation
Core Container
Beans
Core
Context
Expression Language
Test

Project setup

Software requirements:

- JDK (Java Development Kit)
- Apache Tomcat Server (https://tomcat.apache.org/download-90.cgi)
- Eclipse Or Intellij, IDE for JAVA EE developer (https://www.eclipse.org/downloads/packages/release/2020-12/r/eclipse-ide-enterprise-java-developers)
- Note: We will use Maven project to download the required library/dependencies (i.e. avoid adding the dependent jar/s manually)

Project Setup With XML Based configurations:

- Demo Hello-world Spring Applications
- XML based for learning purpose only, preferred is java based and next session onwards, will use Java annotations based only

Day - 2

Spring Container

- Create and manage the objects (Inversion of control)
- Inject object’s dependencies (Dependency Injection)

Introduction to Java based annotations

Project Setup:

- Auto wiring with Java annotations based configurations
- Hello world Spring Application with Java based configurations
- Going forward, will use Java based annotations

Day - 3

CREST DATA SYSTEMS

- → Introduction to Spring Bean
- → Bean Life Cycle
- → Bean Scopes

CREST
DATA SYSTEMS

What is spring bean

- A spring bean is simply a Java object.
- When Java objects are created by the Spring Container, then Spring refers to them as "Spring Beans".
- Spring Beans are created from normal Java classes .... just like Java objects.
- In summary, whenever you see "Spring Bean", just think Java object.

Below is the definition from spring documentation

In Spring, the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and managed by a Spring IoC container. Otherwise, a bean is simply one of many objects in your application. Beans, and the dependencies among them, are reflected in the configuration metadata used by a container.

- In Spring, the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container.

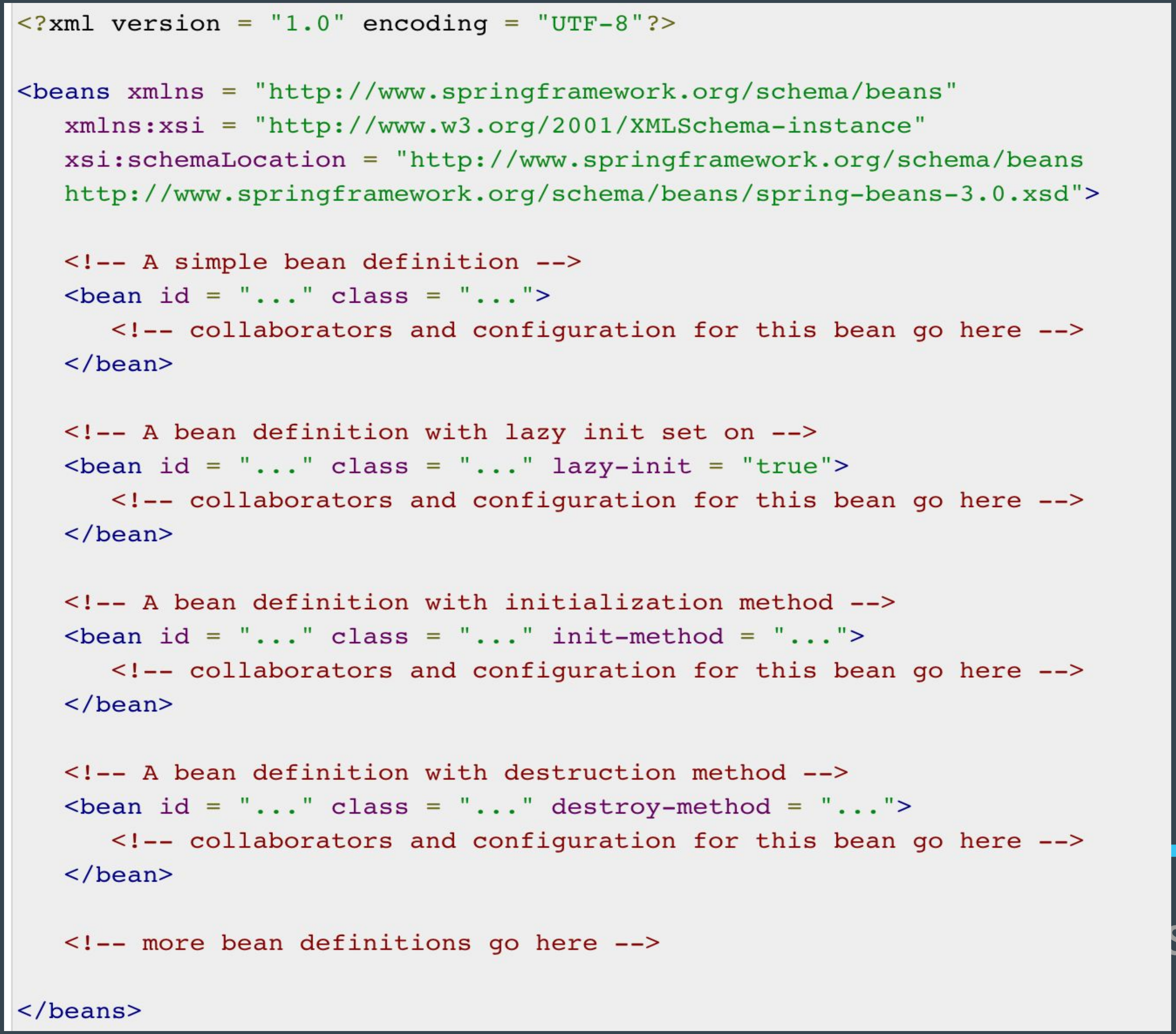
- Simply put, Inversion of Control (IoC) is a process in which an object defines its dependencies without creating them. This object delegates the job of constructing such dependencies to an IoC container.

- A Spring Bean represents a POJO component performing some useful operation. All Spring Beans reside within a Spring Container also known as IOC Container.

- The Spring Framework is transparent and thereby hides most of the complex infrastructure and the communication that happens between the Spring Container and the Spring Beans.

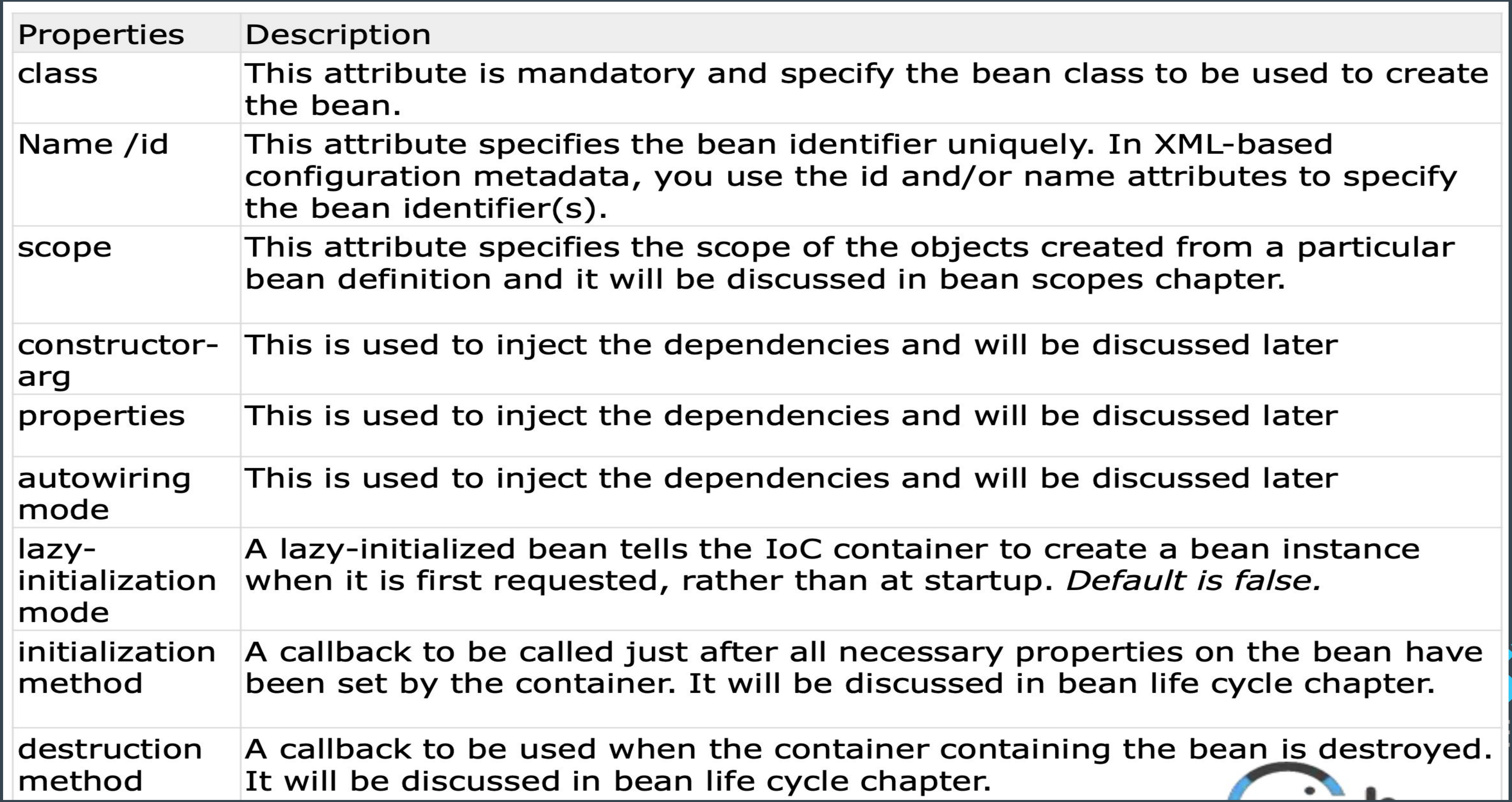
CREST DATA SYSTEMS

Bean Definition



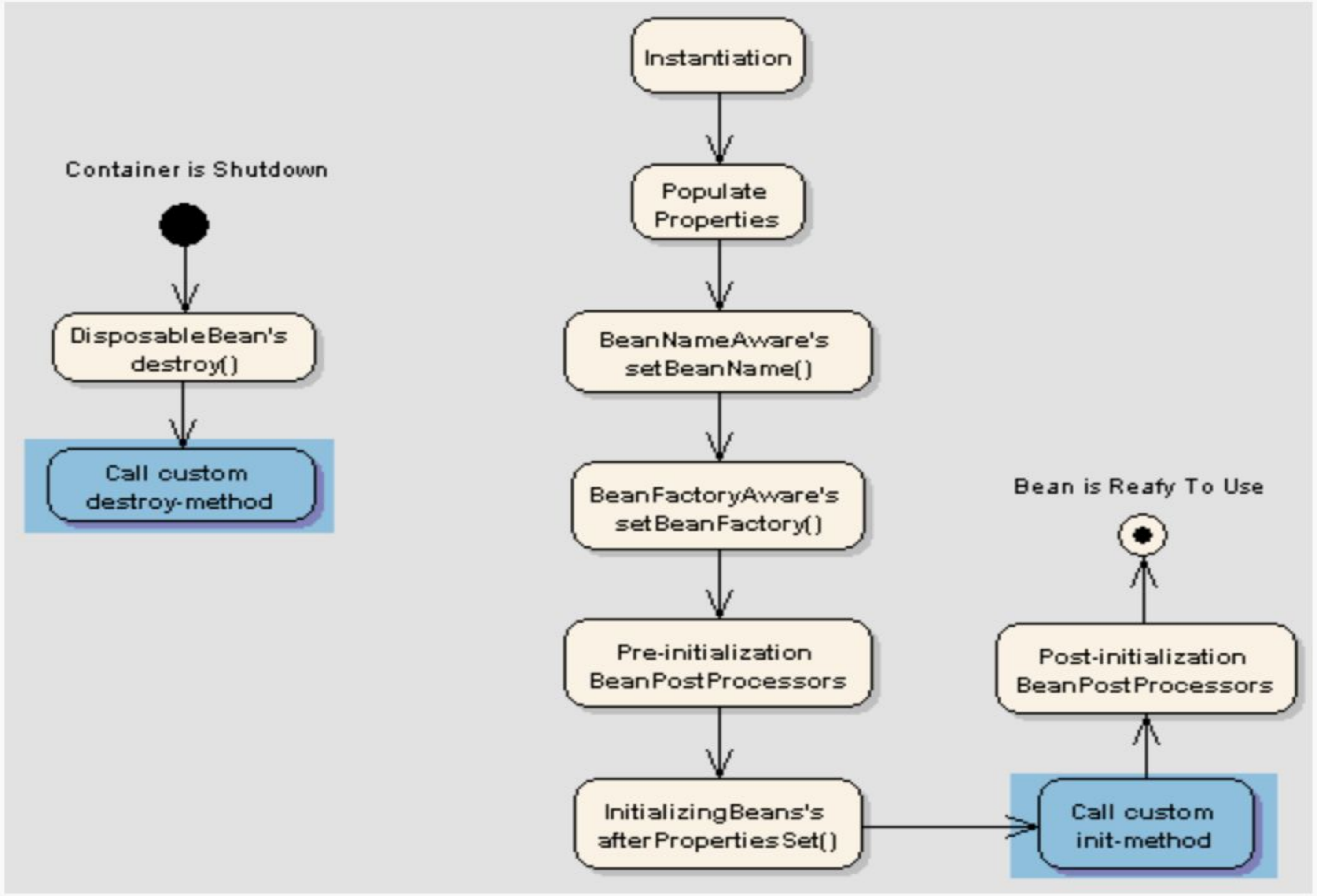
```xml
<?xml version = "1.0" encoding = "UTF-8"?>

<beans xmlns = "http://www.springframework.org/schema/beans"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation = "http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

   <!-- A simple bean definition -->
   <bean id = "..." class = "...">
      <!-- collaborators and configuration for this bean go here -->
   </bean>

   <!-- A bean definition with lazy init set on -->
   <bean id = "..." class = "..." lazy-init = "true">
      <!-- collaborators and configuration for this bean go here -->
   </bean>

   <!-- A bean definition with initialization method -->
   <bean id = "..." class = "..." init-method = "...">
      <!-- collaborators and configuration for this bean go here -->
   </bean>

   <!-- A bean definition with destruction method -->
   <bean id = "..." class = "..." destroy-method = "...">
      <!-- collaborators and configuration for this bean go here -->
   </bean>

   <!-- more bean definitions go here -->

</beans>
```

Bean Properties



Properties	Description
class	This attribute is mandatory and specify the bean class to be used to create the bean.
Name /id	This attribute specifies the bean identifier uniquely. In XML-based configuration metadata, you use the id and/or name attributes to specify the bean identifier(s).
scope	This attribute specifies the scope of the objects created from a particular bean definition and it will be discussed in bean scopes chapter.
constructor-arg	This is used to inject the dependencies and will be discussed later
properties	This is used to inject the dependencies and will be discussed later
autowiring mode	This is used to inject the dependencies and will be discussed later
lazy-initialization mode	A lazy-initialized bean tells the IoC container to create a bean instance when it is first requested, rather than at startup. *Default is false.*
initialization method	A callback to be called just after all necessary properties on the bean have been set by the container. It will be discussed in bean life cycle chapter.
destruction method	A callback to be used when the container containing the bean is destroyed. It will be discussed in bean life cycle chapter.

Bean Life Cycle



Instantiation
Populate Properties
BeanNameAware's setBeanName()
BeanFactoryAware's setBeanFactory()
Pre-initialization BeanPostProcessors
InitializingBeans's afterPropertiesSet()
Call custom init-method
Post-initialization BeanPostProcessors
Bean is Reafy To Use
Container is Shutdown
DisposableBean's destroy()
Call custom destroy-method

Spring Bean Life Cycle

Bean Scope

Scope	Description
singleton	This scopes the bean definition to a single instance per Spring IoC container (default).
prototype	This scopes a single bean definition to have any number of object instances.
request	This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.
session	This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.
global-session	This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

CREST DATA SYSTEMS

Initialization Callbacks

```
<bean id = "exampleBean" class = "examples.ExampleBean"
init-method = "init"/>
```

```
public class ExampleBean implements InitializingBean {
   public void afterPropertiesSet() {
      // do some initialization work
   }
}
```

Destruction Callbacks

```
<bean id = "exampleBean" class = "examples.ExampleBean"
destroy-method = "destroy"/>
```

```
public class ExampleBean implements DisposableBean {
   public void destroy() {
      // do some destruction work
   }
}
```

Reference

- https://www.tutorialspoint.com/spring/spring_bean_scopes.htm
- https://www.tutorialspoint.com/spring/spring_bean_life_cycle.htm
- https://examples.javacodegeeks.com/spring-bean-life-cycle-exampl
- https://www.tutorialspoint.com/spring/spring_bean_definition_inheritance.htm
- https://www.tutorialspoint.com/spring/spring_injecting_inner_beans.htm
- https://www.tutorialspoint.com/spring/spring_injecting_collection.htm

Day - 4

Understanding Spring Web MVC

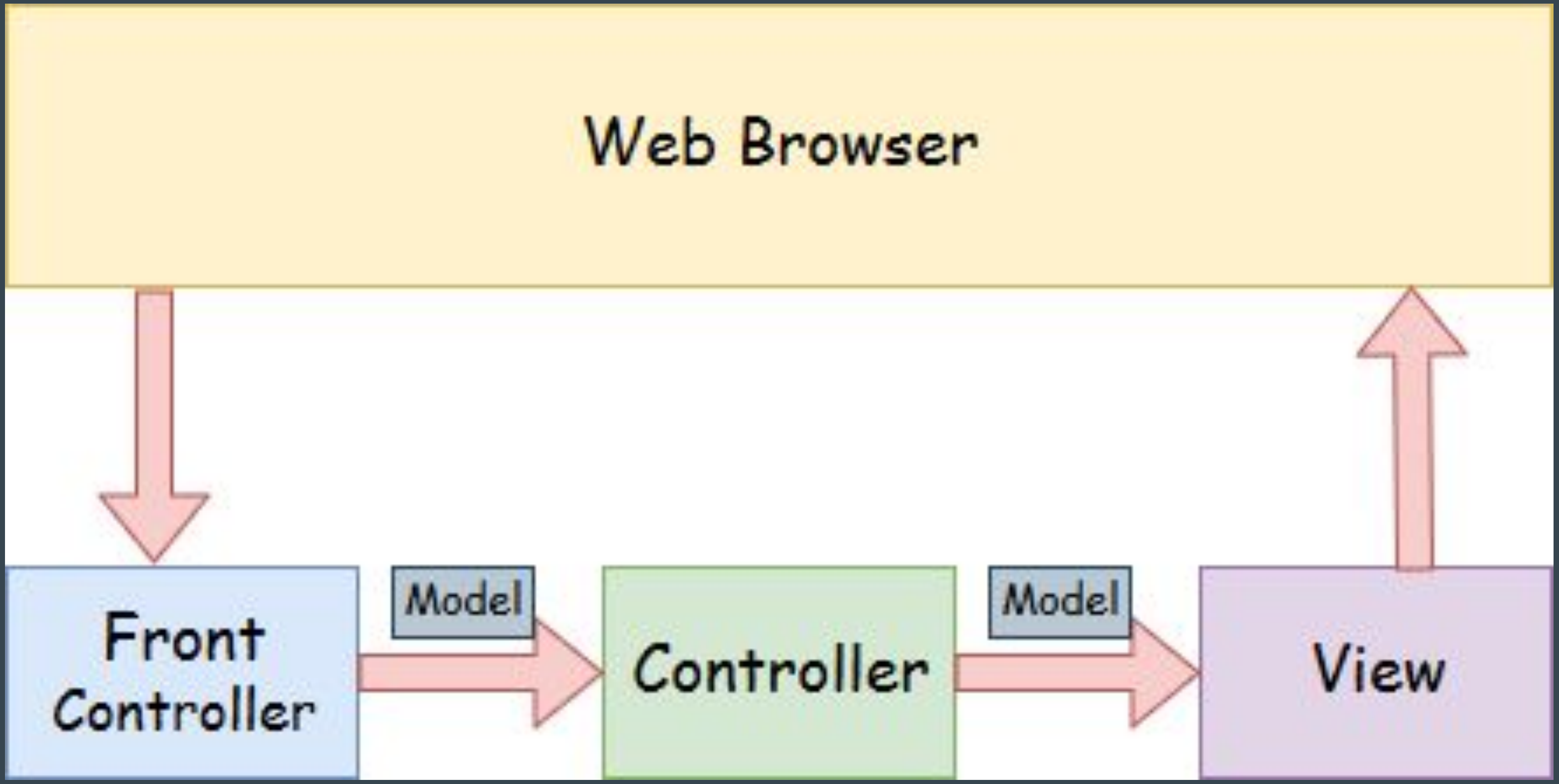
- This module is the first one built by the Spring Team to support the web applications in Spring Framework.
- This module uses the Servlet API as its foundation so these web applications must follow the Servlet Specification and be deployed into servlet containers.
- The Spring Web MVC module was developed using the front controller pattern. Under the hood, there is the main servlet in Spring called DispatcherServlet. This servlet will redirect through an algorithm to do the desired work.
- It enables developers to create amazing web applications on the Java platform.
- Also, the framework allows developers to build REST APIs with this module.

CREST DATA SYSTEMS

Spring Web MVC Project Set-up

Demo

- Set-up: IDE, Maven project, Tomcat server
- Spring web application - Hello World


Web Browser
Front Controller
Model
Controller
Model
View

Day - 5

Spring Web MVC

- Spring MVC with various HTTP request/responses:
- Basic HTTP concepts: e.g. GET v/s POST, Status codes e.g. 200/3xx/4xx/5xx
- Request/response
- Path/query parameters, payload
- Request/response HEADER

CREST DATA SYSTEMS

Spring Boot Training

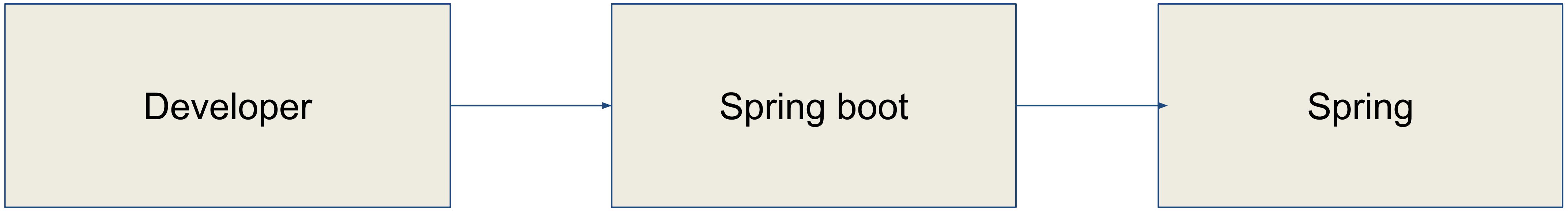
Day - 6

CREST
DATA SYSTEMS

Introduction to Spring Boot

- Spring boot architecture, dispatcher servlet
- Setup spring boot Web project
- Spring boot Web, demo restful service using spring boot starters

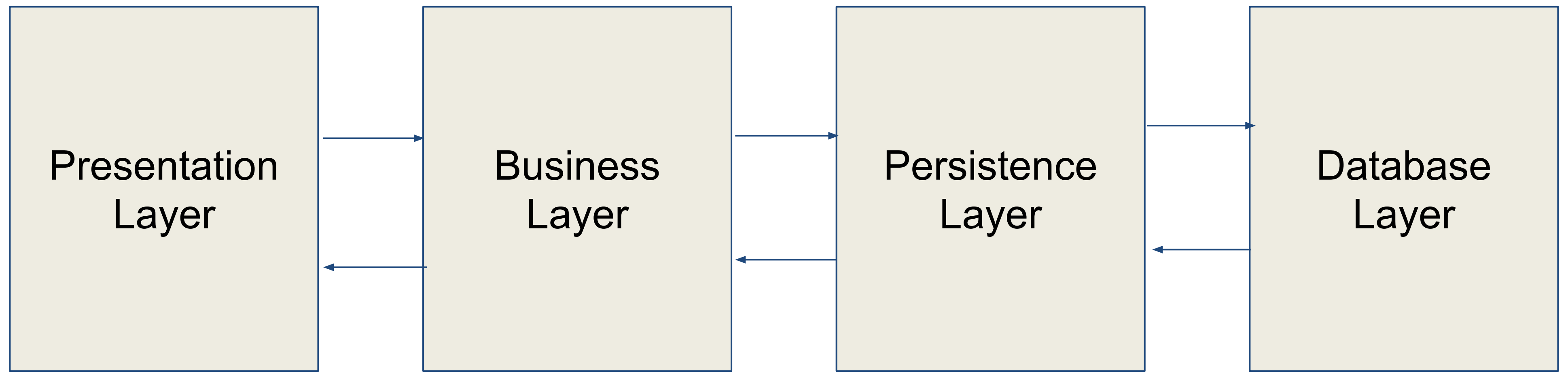
Why Spring boot?



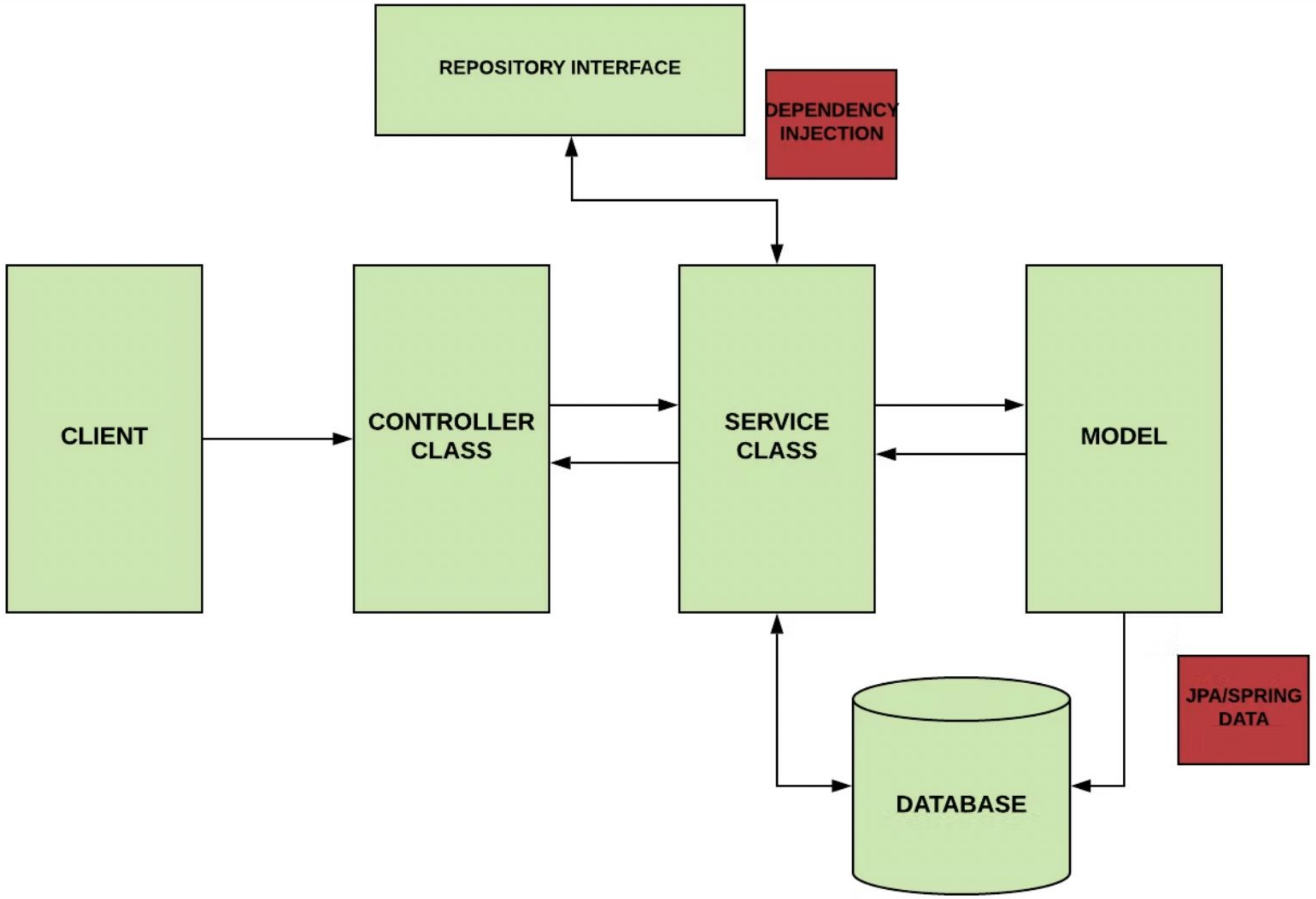
Key points

> Starter Dependencies
> Embedded Server
> Metrics and health check
> Automatic configuration

Architecture of Spring boot

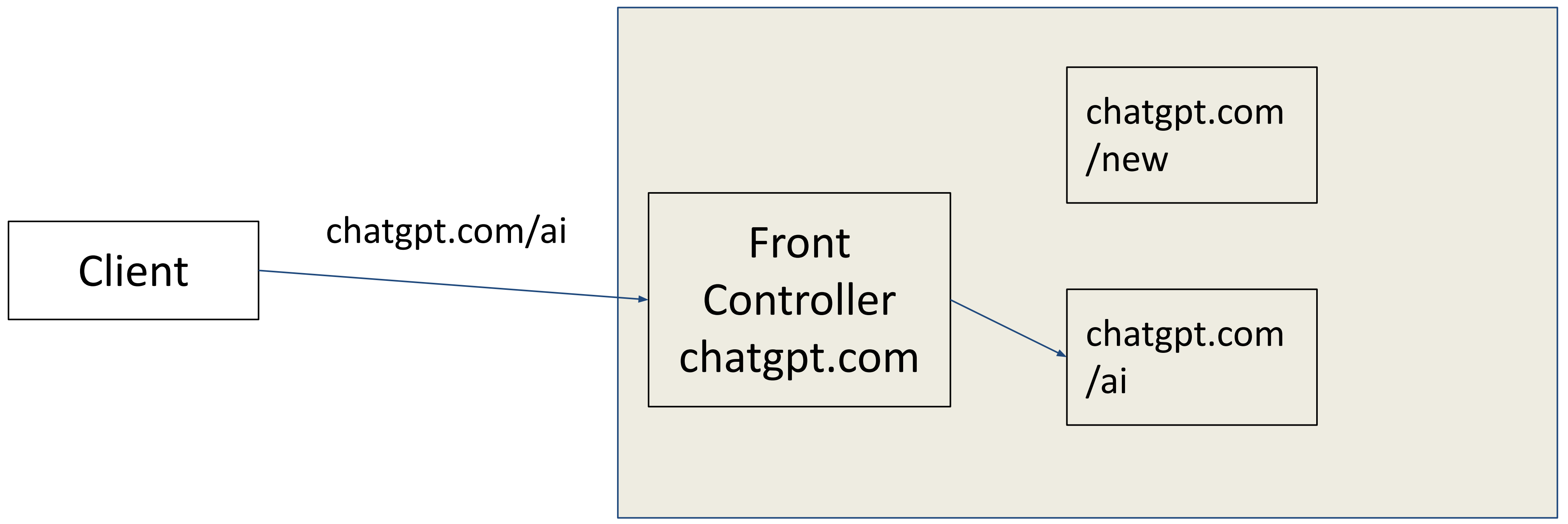


Workflow of Spring boot



Dispatcher Servlet

> Main/Front Controller



Hands-On

- Create a spring boot project, in which user will send json file and in response, the content of the file must be received.
- Create a spring boot project, in which user will send image and in response, the image must be received.

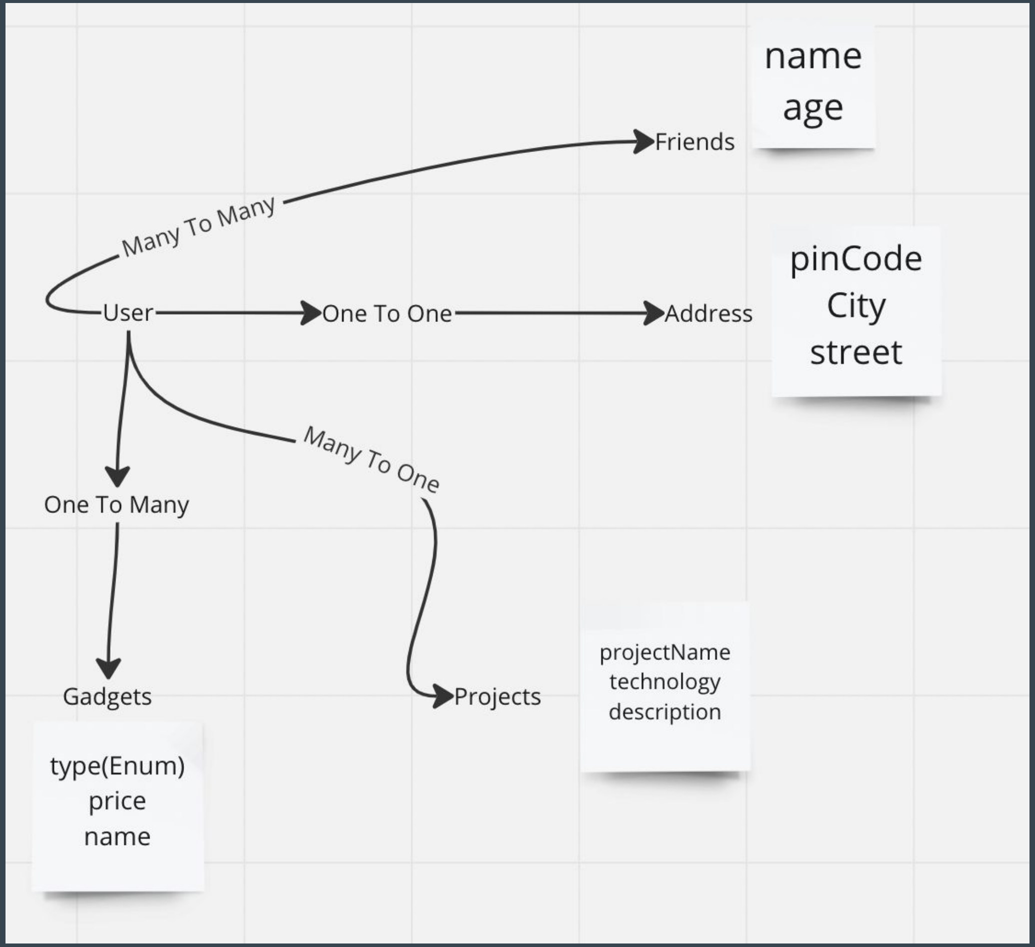
Day - 7

Spring Boot

- CRUD Operations using persistence layer (demo using in memory/H2 database)
- JSON payload/response

Hands-on

Create CRUD operation for User and create database schema



Day - 8

Spring Boot

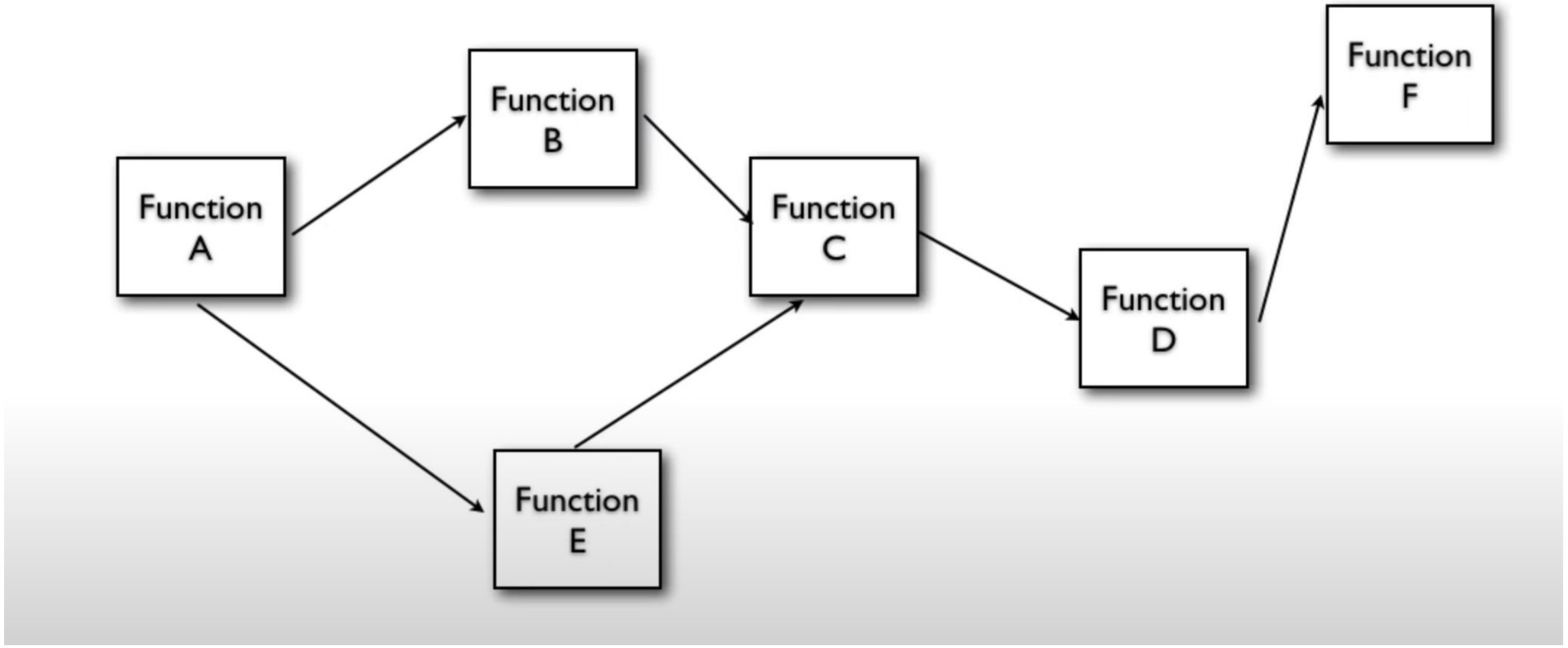
- Pagination overview
- GET all/listing operations with server side pagination/sorting/filtering

Day - 9

Understanding AOP

- Spring Boot Understanding AOP Concepts and Terminology
- AOP demo

Functional Programming



Object Oriented Programming

Object A
Variables
Methods

Object B
Variables
Methods

Object C
Variables
Methods

Common Functionality

Object A
Variables
Methods
log()

Object B
Variables
Methods
log()

Object C
Variables
Methods
log()

Possible Solution

Object A
Variables
Methods
Logger.log()

Object B
Variables
Methods
Logger.log()

Object C
Variables
Methods
Logger.log()

Logger
log()

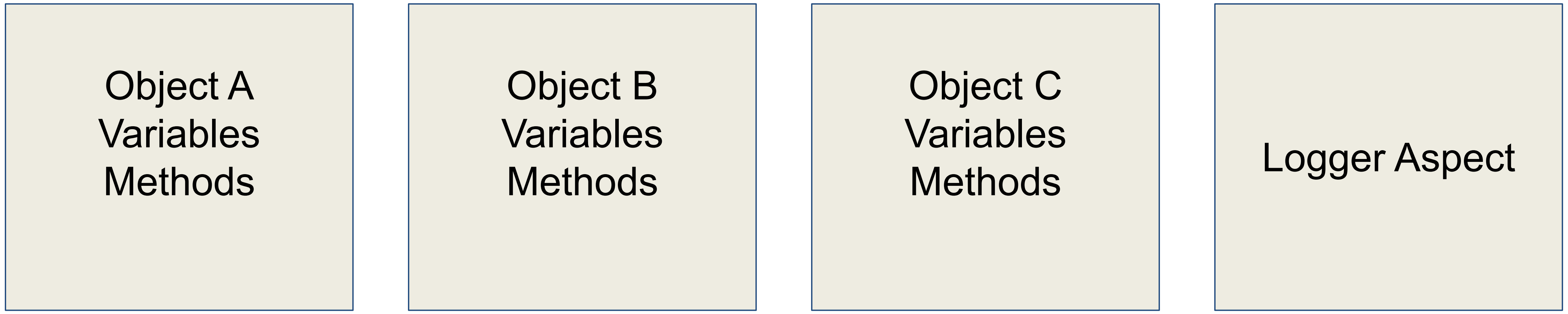
Problems

- Too many relationships to the Crosscutting objects
- Code is required
- Cannot be changed all at once

Other concerns

- Logging
- Transactions
- Security
- Caching
- Monitoring

AOP


Object A
Variables
Methods
Object B
Variables
Methods
Object C
Variables
Methods
Logger Aspect

Hands-On

- Print log message with classname, methodname, arguments before method call and after method call
- Print log message for specific class and specific method

Day - 10

Exception Handling

- Implementing Exception Handling

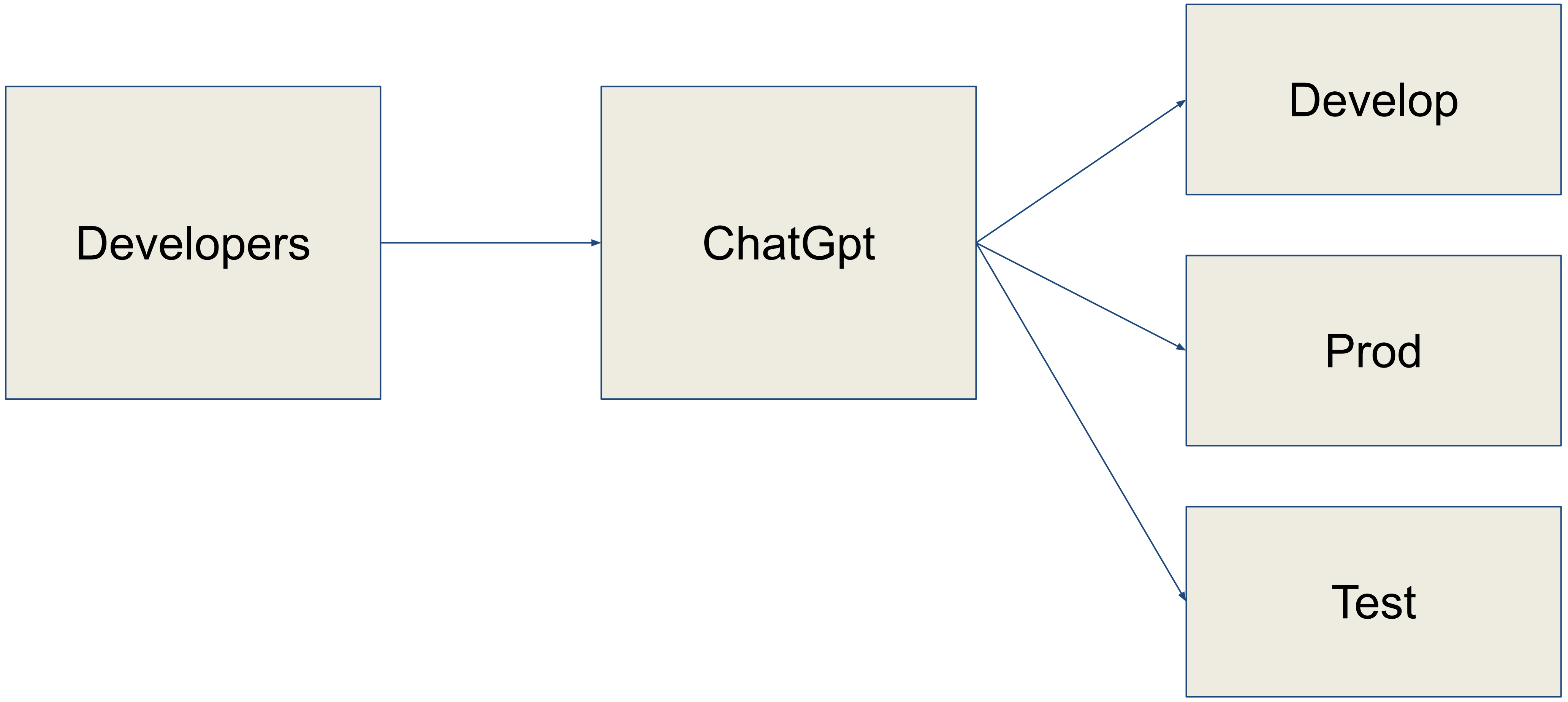
Day - 11

CREST DATA SYSTEMS

Externalized Configs

- Profiles
- Externalized configurations

Profiles and Externalized Configuration



Day - 12

Testing

- Testing in Spring boot
- Two type of testing
  1. Unit testing
     a. Use to test methods
  2. Integration testing
     a. Use to test controller/APIs
- Testing is used to test application for expected behaviour
- Multiple libraries available for testing like,
  - Junit
  - AssertJ

Annotations for test cases

- @springboottest
- @test
- @mock
- @beforeEach
- @beforeAll
- @AfterEach
- @AfterAll

Methods for assertion

- assertEquals
- assertThrows
- assertFalse
- assertDoesNotThrows
- and many more…

MockMvc

Used to perform API testing

ex.

```
mockMvc.perform(MockMvcRequestBuilders.post("/api/v1/user”)
                .header("authorization", bearToken)
                .contentType(MediaType.APPLICATION_JSON)
                .content(content)
        )
        .andExpect(status().isCreated())
        .andExpect(MockMvcResultMatchers.jsonPath("$.name").value("name"))
        .andExpect(MockMvcResultMatchers.jsonPath("$.alias").value("alias"));
```

Hands-On

Write Unit test and Integration test, positive and negative scenario test cases for User model we have used in Day-7 training

CREST DATA SYSTEMS

Spring Boot

- Event Listeners

Events

CREST DATA SYSTEMS

Exchange of information

Method calls

Events

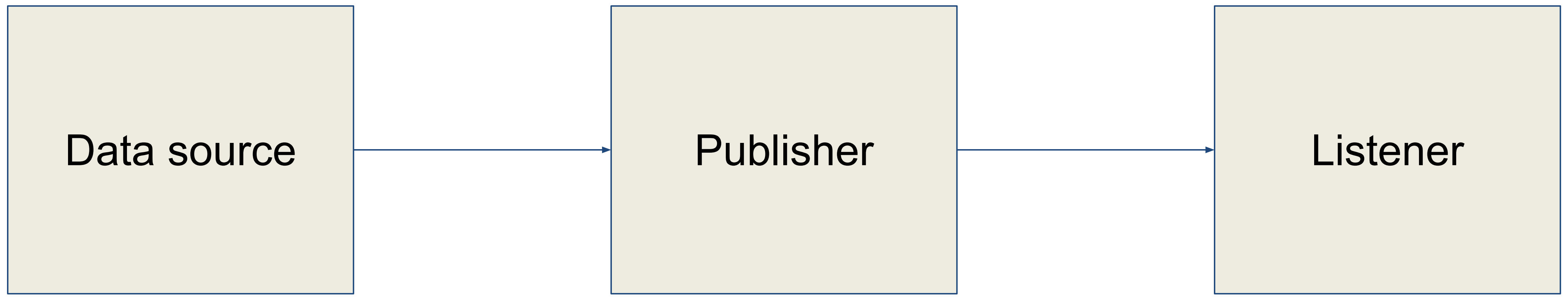
What is event?

- An event, in a computing context, is an action or occurrence that can be identified by a program and has significance for software.

Types of Events

- Application Events
  - User actions
  - Logging in
  - Application Configuration
- System Events
  - System environment
  - Hardware
  - System clock
  - System memory

Steps for creating event



Hands-On

> Create a ecommerce application and create a service to add product in the cart and make that code asynchronous using events.

From here...

Self study, explore more:

- Spring Boot File-upload
- Spring Boot Logging
- View Templates (e.g. Thymeleaf)
- Spring Boot JPA, Spring Boot JDBC
- Spring Boot Interceptor, Servlet Filters
- Spring Boot Internationalization
- Spring Boot Runners
- Spring Boot Authorization, Authentication, OAuth2 with JWT
- Spring Boot Scheduling
- Spring Boot Flyway Database
- Spring Boot Rest Template
- Spring Boot Swagger Documentation
- Spring Boot Actuator, Spring Cloud,
- Basic Docker concepts

CREST DATA SYSTEMS