Spring 5.x

ANIL JOSEPH

Presented by Anil Joseph(anil.jos@gmail.com)

Agenda

Introduction to Spring	Spring Features & Components	Inversion of Control (Dependency Injection)
Aspect Oriented Programming(*Spring AOP*)	Spring JDBC and Transactions	Spring Data
Spring MVC Web (REST Webservices)	Spring Boot (Introduction)	Spring Security

What does Spring offer?

Container
- Contains and Manages the lifecycle and configuration of application objects

Inversion of Control(Dependency Injection)
- Promotes loose coupling

Aspect-oriented programming
- Enables cohesive development by separating application business logic from system services

Data Access Framework
- Spring JDBC: Enables writing clean and simple data access code
- Spring ORM:Integrates with ORM's
- Spring Transaction: Transaction Management
- Spring Data: Simplified API's for JPA

Agenda

Introduction to Spring	Spring Features & Components	Inversion of Control (Dependency Injection)
Aspect Oriented Programming(*Spring AOP*)	Spring JDBC and Transactions	Spring Data
Spring MVC Web (REST Webservices)	Spring Boot (Introduction)	Spring Security

Presented by Anil Joseph(anil.jos@gmail.com)

Software

JDK 1.8

IntelliJ

Maven

Database (Derby)

Presented by Anil Joseph(anil.jos@gmail.com)

What is Spring?

- Spring is a lightweight enterprise application framework
- Open source and addresses the complexity of enterprise application development
- Its simple, promotes testing and loose coupling
- Not limited to server-side development
  - Create standalone, server, mobile applications
  - Cloud ready
- Integration
  - Integrates with existing frameworks like Hibernate, Struts, iBatis, hessain, etc

Presented by Anil Joseph(anil.jos@gmail.com)

Spring history

- Spring was originally written by Rod Johnson and described in his book "*Expert One-on-One: J2EE Design and Development".*
- The first milestone release was in March 2004
- Major Versions
  - Spring 2.0 in October 2006
  - Spring 2.5 in November 2007
  - Spring 3.0 in December 2009
  - Spring 4.0 in July 2015
  - **Spring 5.0 in July 2016**

What does Spring offer?

Container

- Contains and Manages the lifecycle and configuration of application objects

Inversion of Control(Dependency Injection)

- Promotes loose coupling

Aspect-oriented programming

- Enables cohesive development by separating application business logic from system services

Data Access Framework

- Spring JDBC: Enables writing clean and simple data access code
- Spring ORM:Integrates with ORM's
- Spring Transaction: Transaction Management
- Spring Data: Simplified API's for JPA

What does Spring offer?

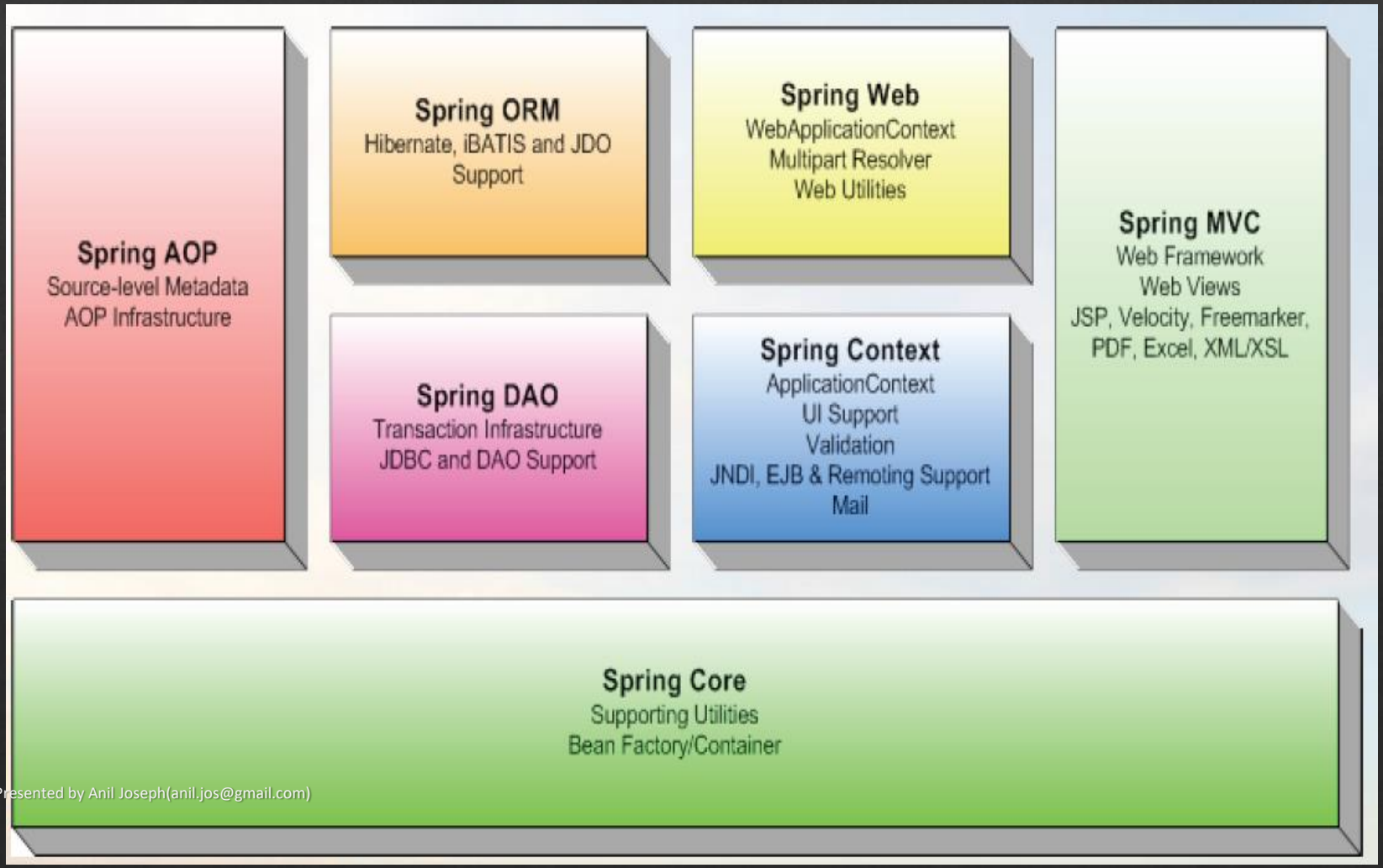
MVC Framework

- MVC framework for web and portal applications
- RESTful Web Services

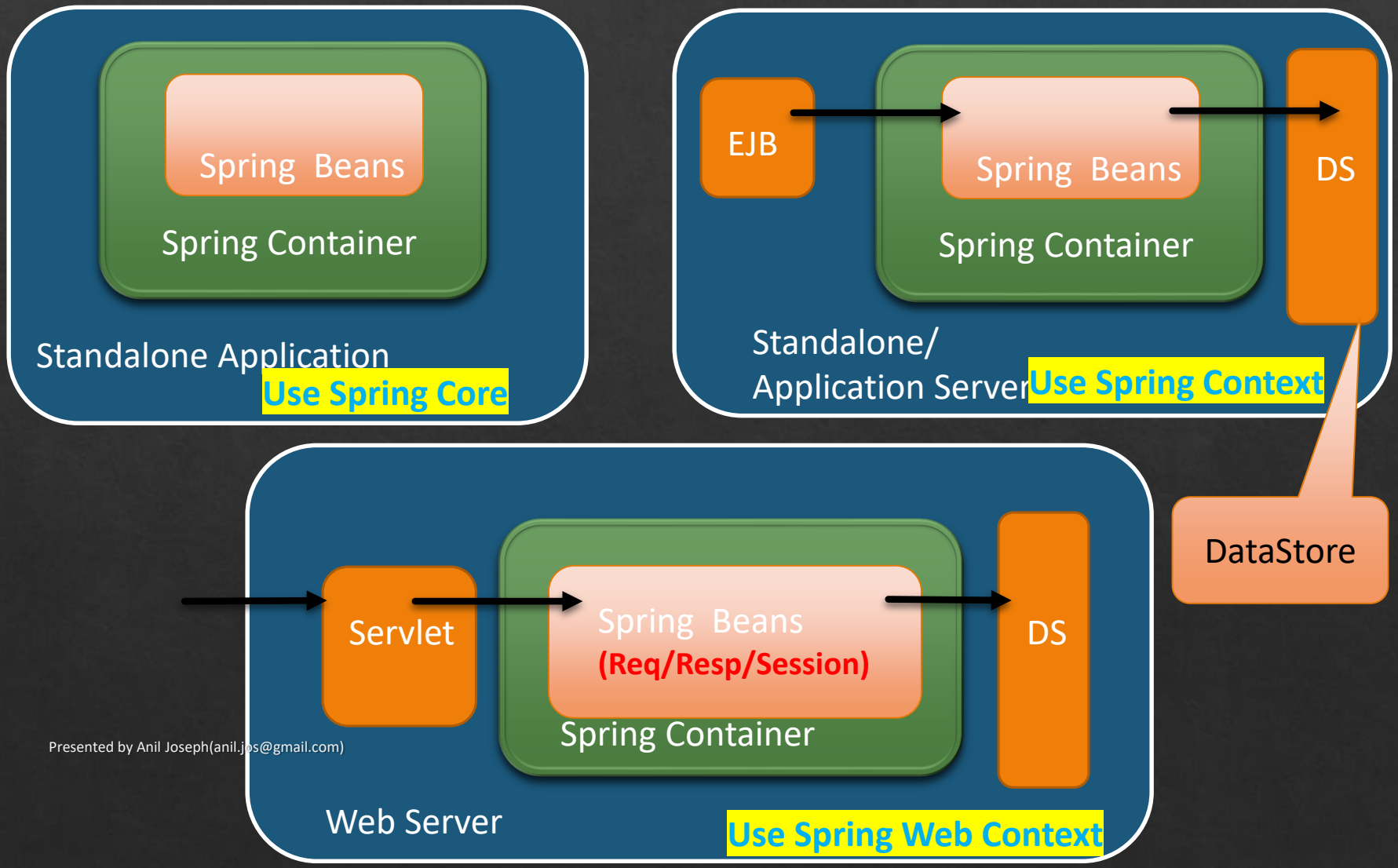
Spring related projects

- Spring Security
- Spring Boot
- Spring WebFlow
- Spring Integration
- Any many more…

Spring Modules



Spring Container



Presented by Anil Joseph(anil.jos@gmail.com)

Demo

Spring Beans

Spring Beans are objects managed by the Spring Container

They follow the Java Beans Specification

Presented by Anil Joseph(anil.jos@gmail.com)

Spring Containers

Containers are the core of spring framework

- They manage the spring beans.

Two types of Containers

- Bean Factory
- Application Context

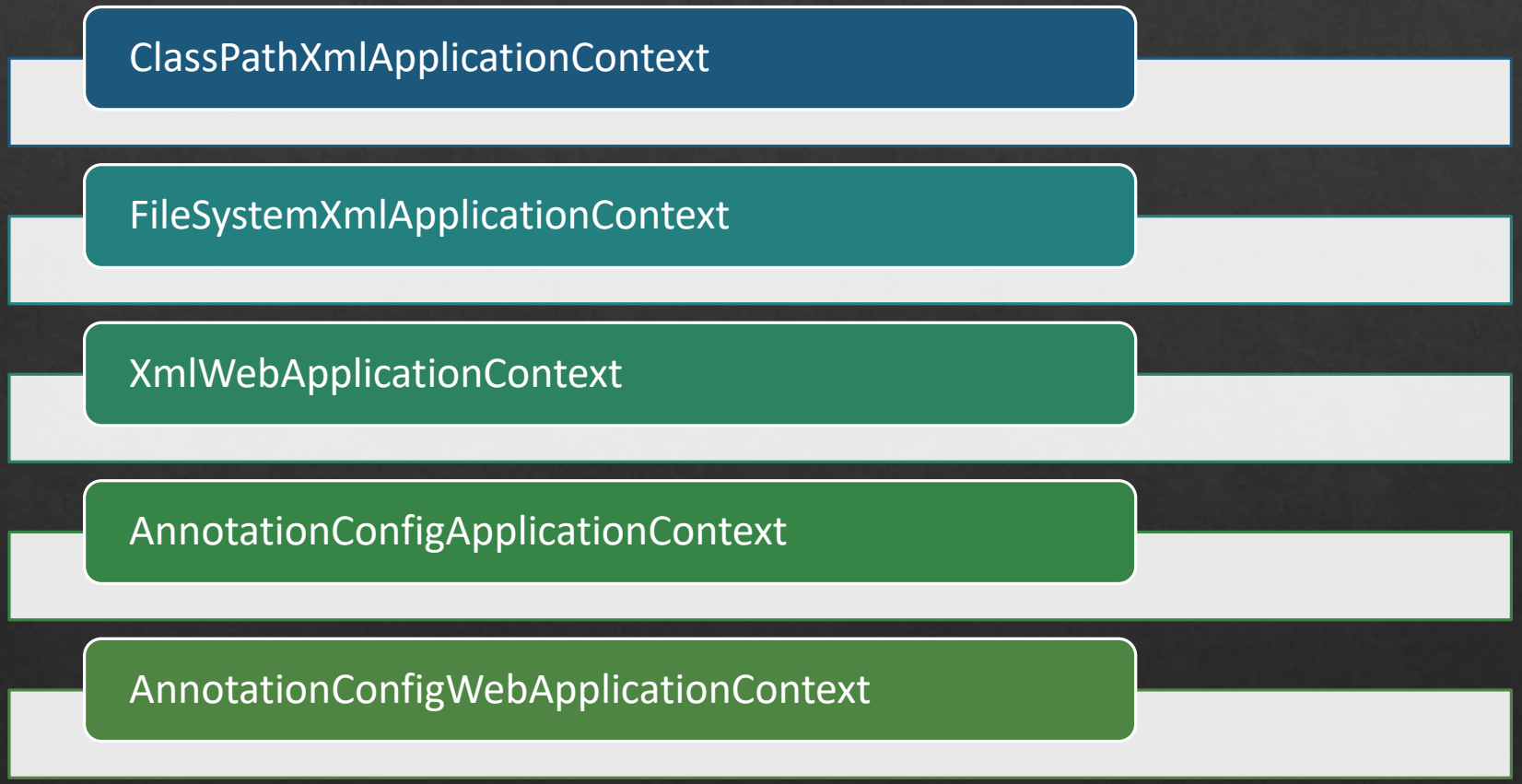
Application Context

Superset of the Bean Factory hence provides all the functionalities of BeanFactory

Additional features for Enterprise or JEE applications.

Supports internationalizations for text messages

Application Context Implementations



Inversion of Control

Principle: *"Don't call us, we'll call you."*

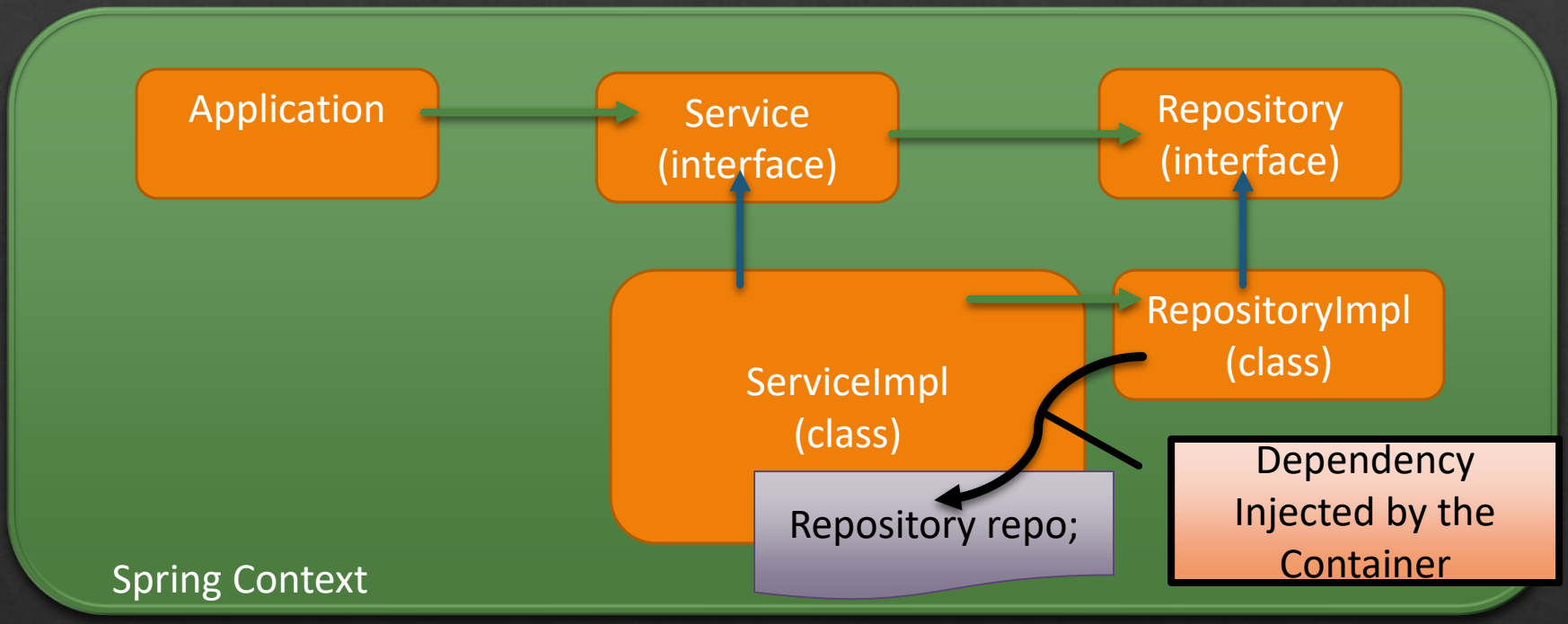
IoC is a principle that is used to wire an application together

Defines how dependencies or object graphs are created.

In Spring, the IoC "flavor" is referred to as ***Dependency Injection***

Presented by Anil Joseph(anil.jos@gmail.com)

IOC



DO Not initialize the references when using Dependency Injection

Repository repo = new RepositoryImpl();

Repository repo = factory.getInstance();

Dependency Injection Mechanisms

Property Injection

- The dependency is injected through a property.

Constructor Injection

- The dependency is injected through a constructor.

Component Scan

A mechanism of scanning a package and its sub packages for spring beans.

The beans get configured with the Spring Context.

The class has to annotated with the @Component annotations or any of types

- @Service
- @Repository
- @Controller

Auto Wiring Annotations

@Autowired

- Marks a constructor, field or setter method as to be autowired by Spring's dependency injection facilities.
- Only one constructor (at max) of any given bean class may carry this annotation.
- Resolves the dependency byType else byName

@Qualifier

- This annotation may be used on a field or parameter as a qualifier for candidate beans when autowiring.
- Used with @Autowired
- Resolves the dependency byName

Wiring with expressions

- Spring 3 introduced the *Spring Expression Language (SpEL)*
- A powerful yet succinct way of wiring values into a bean's properties or constructor arguments using expressions that are evaluated at runtime.
- SpEL advantages,
  - The ability to reference beans by their ID
  - Invoking methods and accessing properties on objects
  - Mathematical, relational, and logical operations on values
  - Regular expression matching
  - Collection manipulation

SpEL Examples

- <property name="count" value="#{5}"/>
  - Sets the value 5 to the count property
- <property name="message" value=" #{‘hello world’}"/>
  - Sets the value “hello world” to the message property
- <property name="capacity" value="#{1e4}"/>
  - Sets the value 10000.0 to the property capacity using the scientific notation
- <property name="enabled" value="#{false}"/>
  - Set the boolean value “false” to the enabled property
- <property name="instrument" value="#{saxophone}"/>
  - Sets the instrument property to the bean with the bean id “saxophone”

SpEL Examples

- <bean id="carl"
  class="com.spring.spring.Instrument">
  <property name="song" value="#{kenny.song}" />
  </bean>
  - Sets the song property to the value returned by the song property of the bean with the id “kenny”
- <property name="song" value="#{songSelector.selectSong()}"/>
  - Sets the song property to the value returned by the method “selectSong”

Lookup Method Example

```
<bean id="service" class="com.ServiceImpl"
    scope="prototype">
</bean>

<bean id="app" class="com.Application">
    <lookup-method name="createService"
    bean="service"/>
</bean>
```

Beans Scope

Singleton

- A single instance created per context

Prototype

- Instances created whenever a bean is requested from the context

Request

- Instance per Http request

Session

- Instance per Http session

Lifecycle Methods

- 3 mechanisms
  - Implement interface InitializingBean & DisposableBean
  - Provide custom initialization and cleanup methods
  - Use the annotations @PostConstruct and @PreDestroy
- Order of invocation
  - Constructor
  - Property Injecion
  - Annotated methods(@PostConstruct and @PreDestroy
  - Interface implementations InitializingBean & DisposableBean
  - Custom methods
- If the scope is prototype the cleanup methods are not invoked.

Aspect-Oriented Programming (AOP)

Aspects enable the modularization of concerns

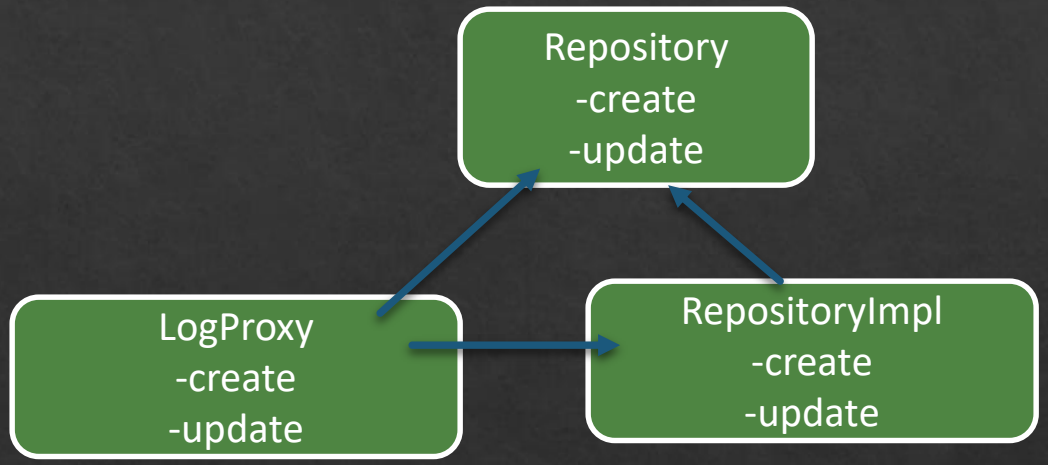
- Transaction management
- Security
- Logging

Such concerns are termed ***crosscutting concerns*** *in AOP literature*

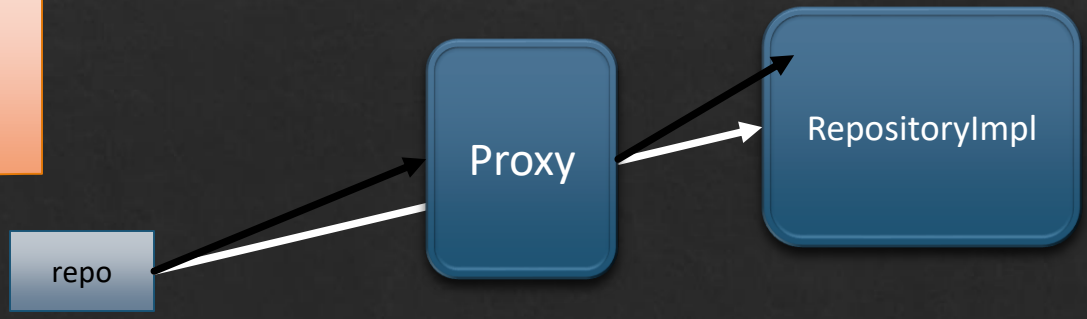
AOP

Using Proxy Design Pattern

- Add Behavior




```
Repository repo = ctx.getBean(..);
repo.create();
repo.update();
```



AOP Concepts

Aspect:

- The unit modularization that cuts across multiple classes.

Join point:

- A point during the execution of a program, such as the execution of a method or the handling of an exception.
- Example methods, constructors, properties etc.

Pointcut:

- A predicate that matches join points.
- Identifies which objects to be injected with the aspect

Advice:

- The set of instructions to execute to implement the aspect
- Action taken by an aspect at a particular join point.

Weaving:

- Defines how the advice is applied to a set of objects.
- This could be compile-time, load-time or run-time.

Types of advice

Before advice

- *Invoked before the method is invoked*

After returning advice

- *Invoked after the method but only if no exceptions are thrown*

After throwing advice

- *Invoked after the method but only if exceptions are thrown*

After (finally) advice

- *Invoked after the method*

Around advice

- *Invoked before and after the method*

Spring JDBC

- The value-add provided by the Spring Framework as a JDBC abstraction.
- Spring JDBC takes care of all the low-level details that can make JDBC such a tedious API to develop with.
- Advantages its provides
  - Open the connection.
  - Prepare and execute the statement.
  - Handles the code of iteration
  - Processing Exceptions
  - Handles transactions
  - Close the connection, statement and resultset.

Presented by Anil Joseph(anil.jos@gmail.com)

Spring JDBC API

JdbcTemplate

- The classic Spring JDBC class

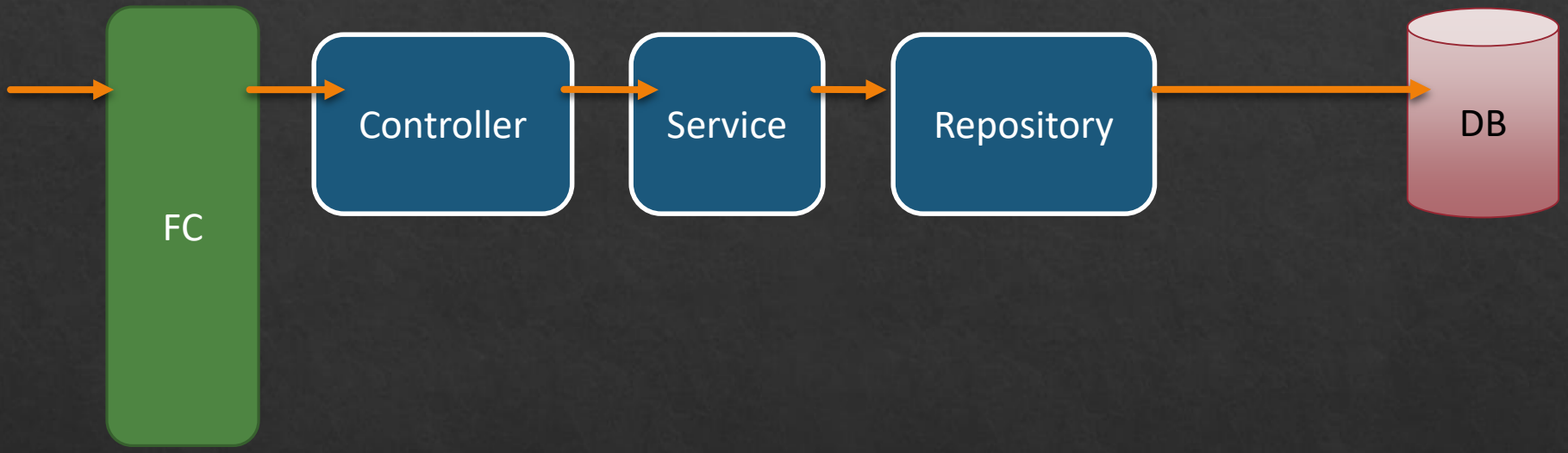
NamedParameterJdbcTemplate

- To provide named parameters instead of the traditional JDBC "?" placeholders.

SimpleJdbcInsert and SimpleJdbcCall

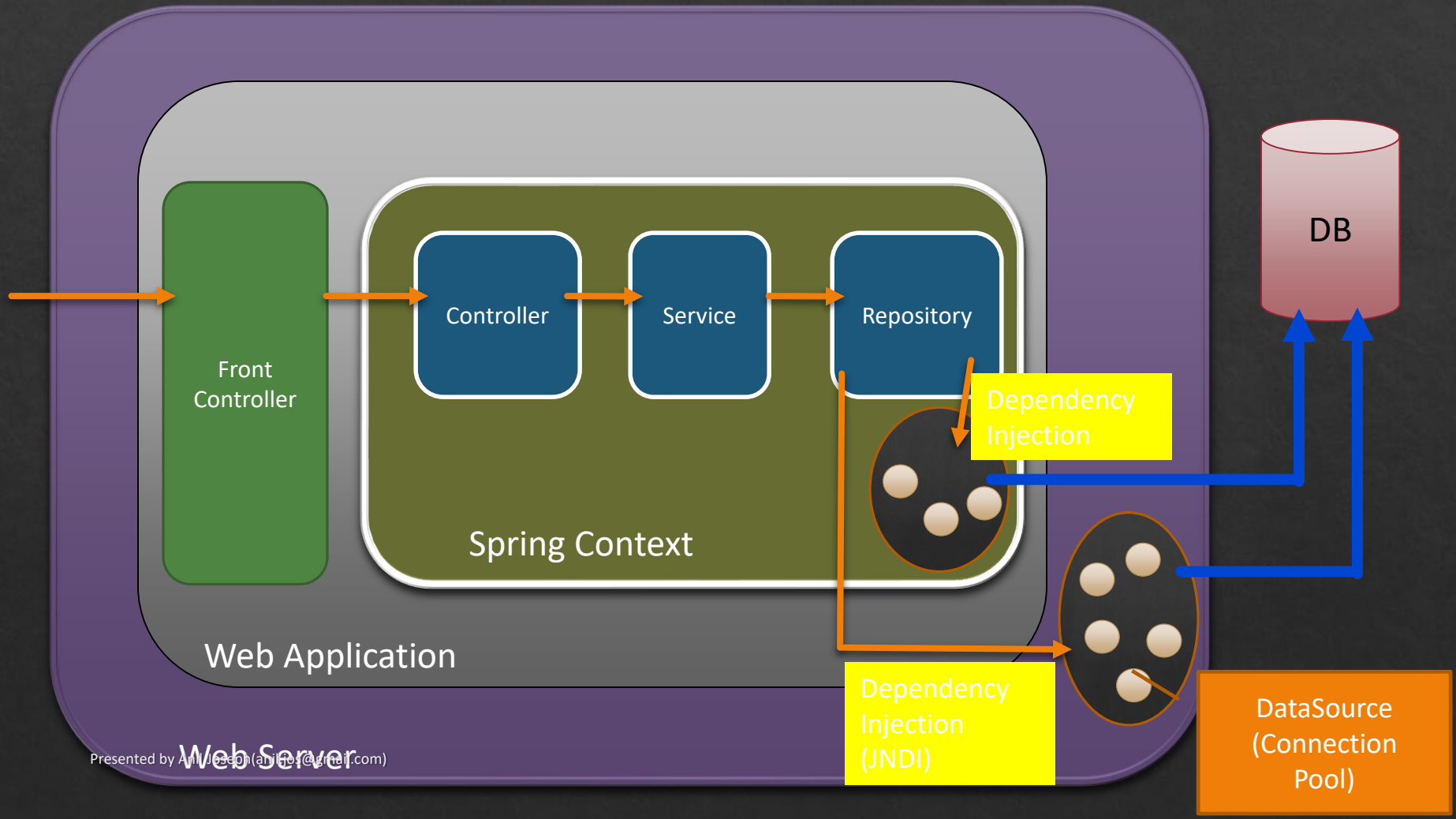
- Optimize database metadata to limit the amount of necessary configuration.

JDBC Application Design

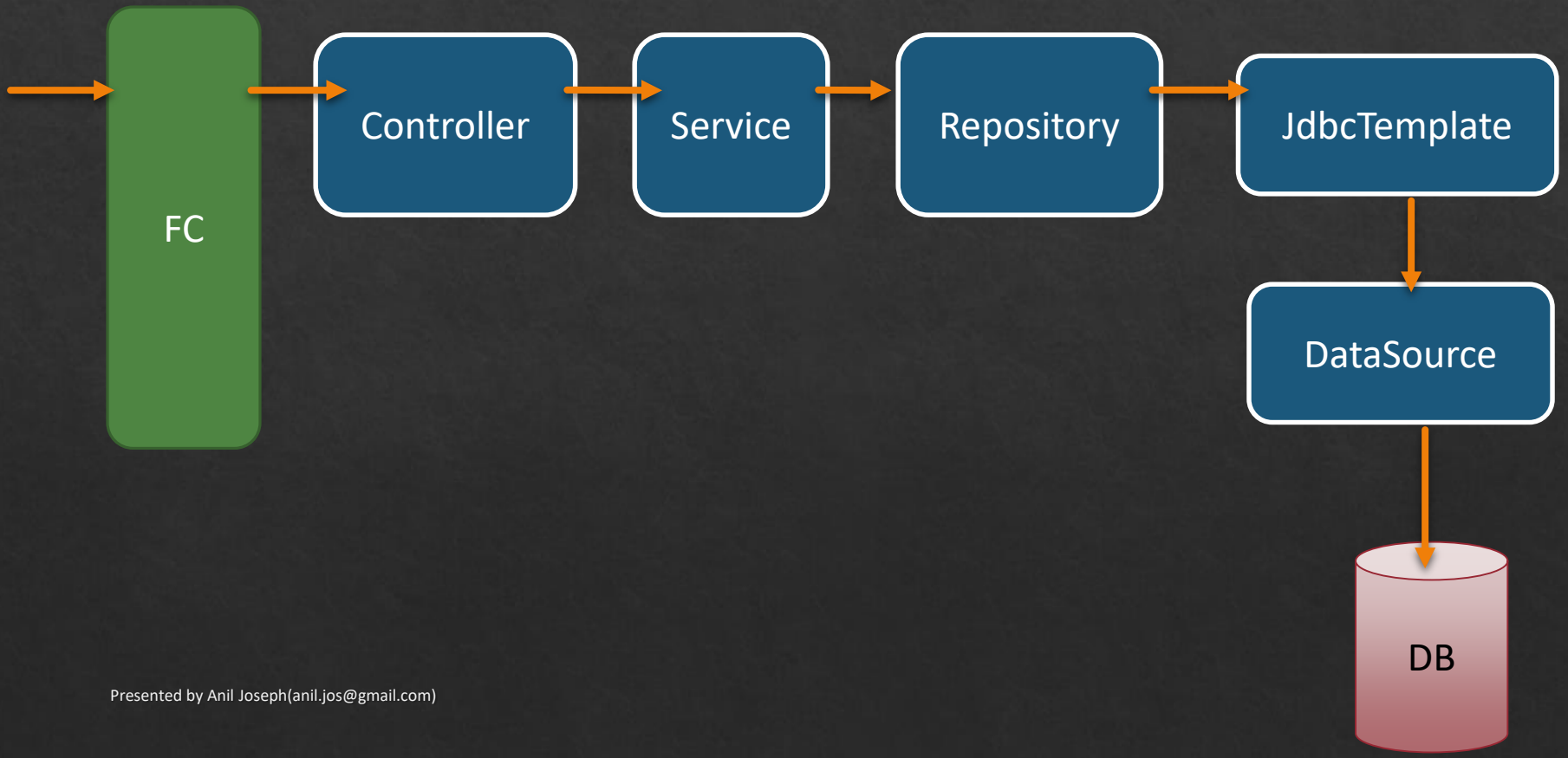


Presented by Anil Joseph(anil.jos@gmail.com)

JDBC Application Design



JDBC Application Design



Presented by Anil Joseph(anil.jos@gmail.com)

Thank You

Presented by Anil Joseph(anil.jos@gmail.com)