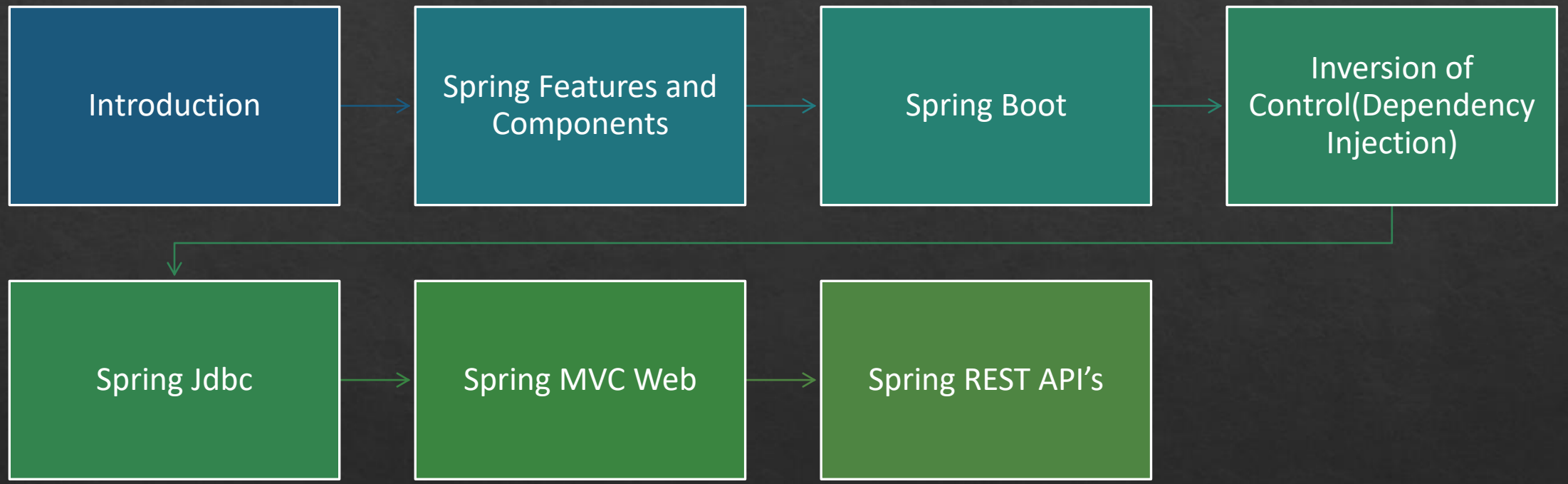
Spring

ANIL JOSEPH

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

Agenda



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

Spring

- Spring is a lightweight enterprise framework
- Open source and address the complexities of enterprise application development
- Simple, promotes TDD and loose coupling.
- Used to create standalone, web, enterprise, and cloud deployed applications
- Integrates with other frameworks like Hibernate, Struts, iBatis, Hessian etc.

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

What Spring offers?

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 for JPA

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

What Spring offers?

MVC Framework

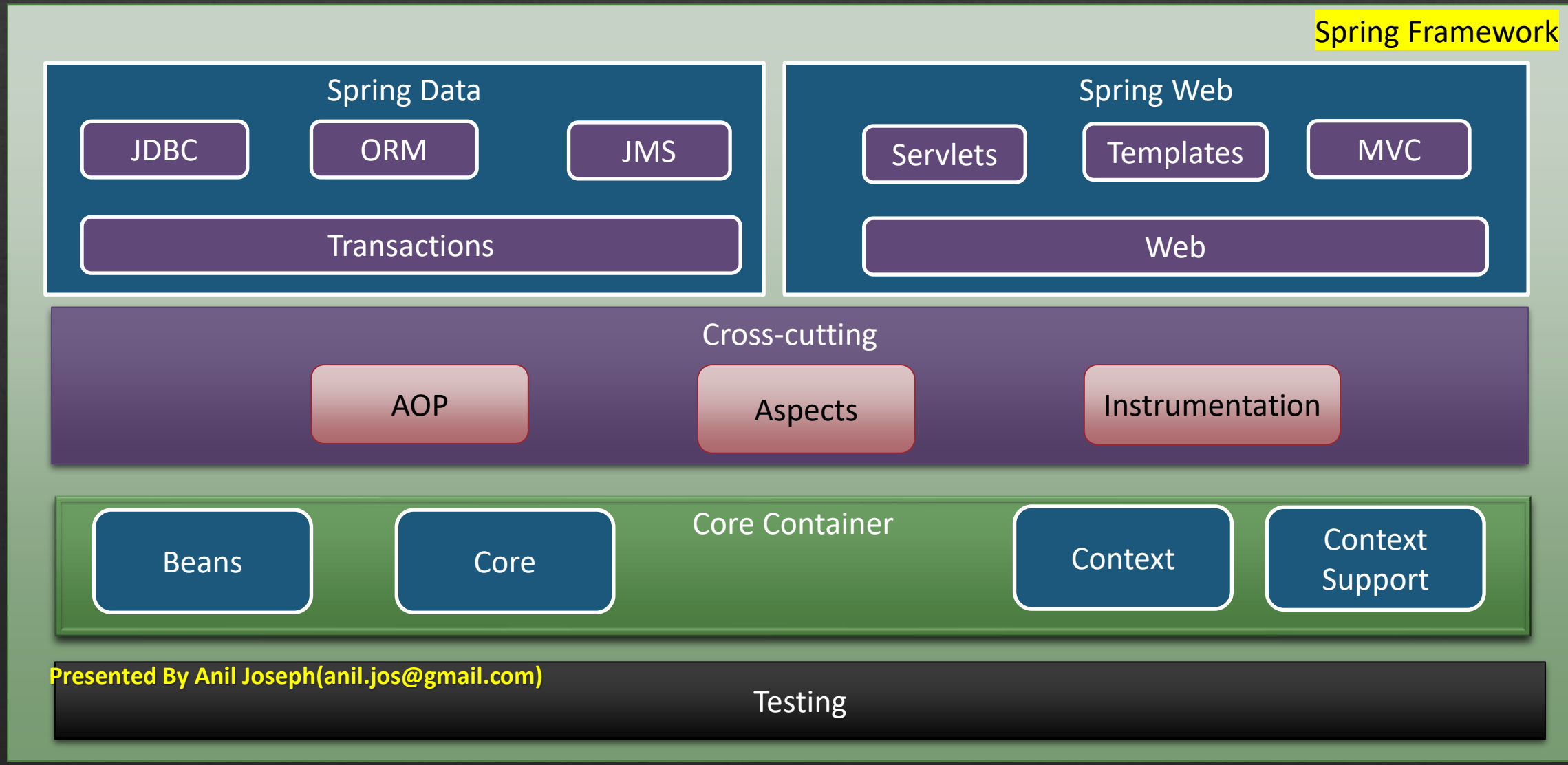
- MVC framework for web and portal applications
- RESTful Web Services

Other Spring Projects

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

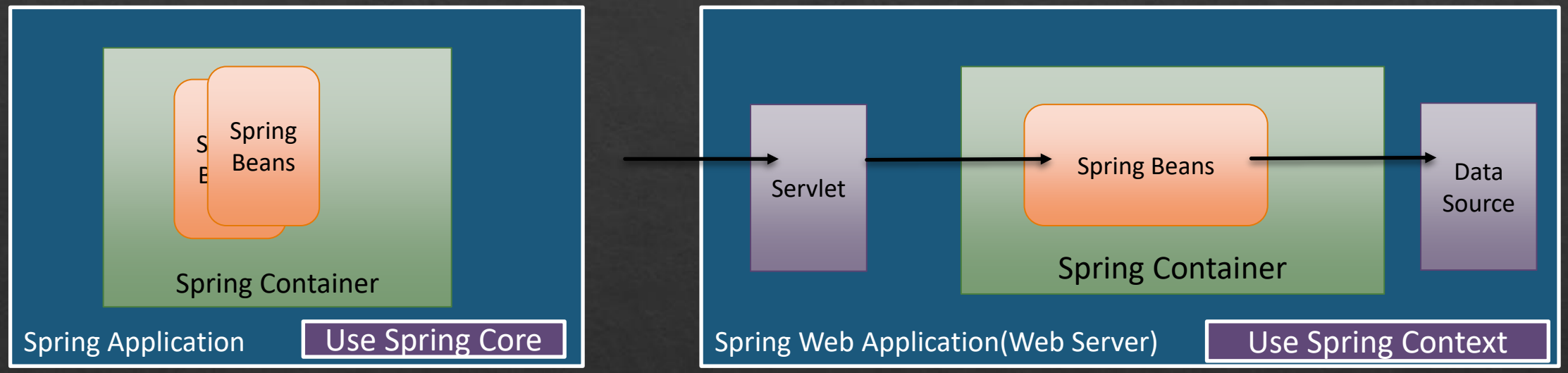
Presented By Anil Joseph(anil.jos@gmail.com)

Spring Modules



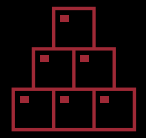
Presented By Anil Joseph(anil.jos@gmail.com)

Spring Container



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

Spring Beans



Spring beans are objects managed by the Spring Container.

The follow the Java Bean 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

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

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

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

Application Context Implementations

ClassPathXmlApplicationContext

- XML configuration in the class path

FileSystemXmlApplicationContext

- XML configuration in the file system

XmlWebApplicationContext

- XML configuration for a web application

AnnotationConfigApplicationContext

- Pure annotation configuration

AnnotationConfigWebApplicationContext

- Pure annotation configuration for web application

Spring Boot

- Provide a radically faster and widely accessible ***getting started experience*** for all Spring development.
- Automatically configure Spring whenever possible
- Opinionated(out of the box)
  - Provides default configuration
- Customizable when requirements diverge from the defaults
- Provide a range of non-functional features
  - Security, Metrics, externalized configuration etc.
- Creates Standalone Applications
- Embed Tomcat, Jetty directly, facilitating deployments on to the cloud
- Based on Maven and Gradle
- No requirement for XML configuration

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

Getting Started

- Spring CLI
  - A command line tool
  - Allows Groovy scripts/ Java Code
  - Very less boilerplate code
- Spring Initializer(https://start.spring.io/)
  - A web interface to generate the Maven or Gradle projects
- Eclipse and IntelliJ plugins
  - Provide templates to create spring boot applications

Spring Boot Application

- @SpringBootApplication
  - The annotation is used on the main class of the application
  - Its an alternative to @Configuration, @ComponentScan & @EnableAutoConfiguration
- @Configuration
  - Indicates that a class declares on or more @Bean methods.
  - @Configuration classes are typically bootstrapped using
    - AnnotationConfigApplicationContext
    - AnnotationConfigWebApplicationContext
- @ComponentScan
  - Provides support parallel with Spring XML's <context:component-scan> element.

Spring Boot Application

- @EnableAutoConfiguration
  - Enable auto-configuration of the Spring Application Context
  - Automatically loads all the beans required by the application
- SpringApplication class
  - The SpringApplication class provides a convenient way to **bootstrap** a Spring application that will be started from a main() method
  - Will create an ***AnnotationConfigApplicationContext*** or ***AnnotationConfigEmbeddedWebApplicationContext***
  - Provides a set of events and listeners

Component Scan

- A mechanism of scanning a package and its sub packages for spring beans.
- The beans get registered with the Spring Context .
- The class has to annotated with the @Component annotations or any of types
  - @Service
  - @Repository
  - @Controller

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

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)

Benefits

- Decoupling: IoC helps in decoupling the application components, making the system more modular and easier to maintain.
- Reusability: Components can be reused across different applications or parts of the same application.
- Testability: Decoupled components can be easily tested independently of each other.
- Manageability: IoC provides a centralized way to manage dependencies, configuration, and lifecycle of objects.

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

IoC in Spring

- In Spring, IoC is implemented using the Spring IoC container.
- The container is responsible for instantiating, configuring, and managing the lifecycle of Spring beans.
- It uses dependency injection (DI) to achieve IoC.

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

Dependency Injection Mechanisms

Property(Setter) Injection

- The dependency is injected through a property.

Constructor Injection

- The dependency is injected through a constructor.

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

Auto-wiring annotations

@Autowired

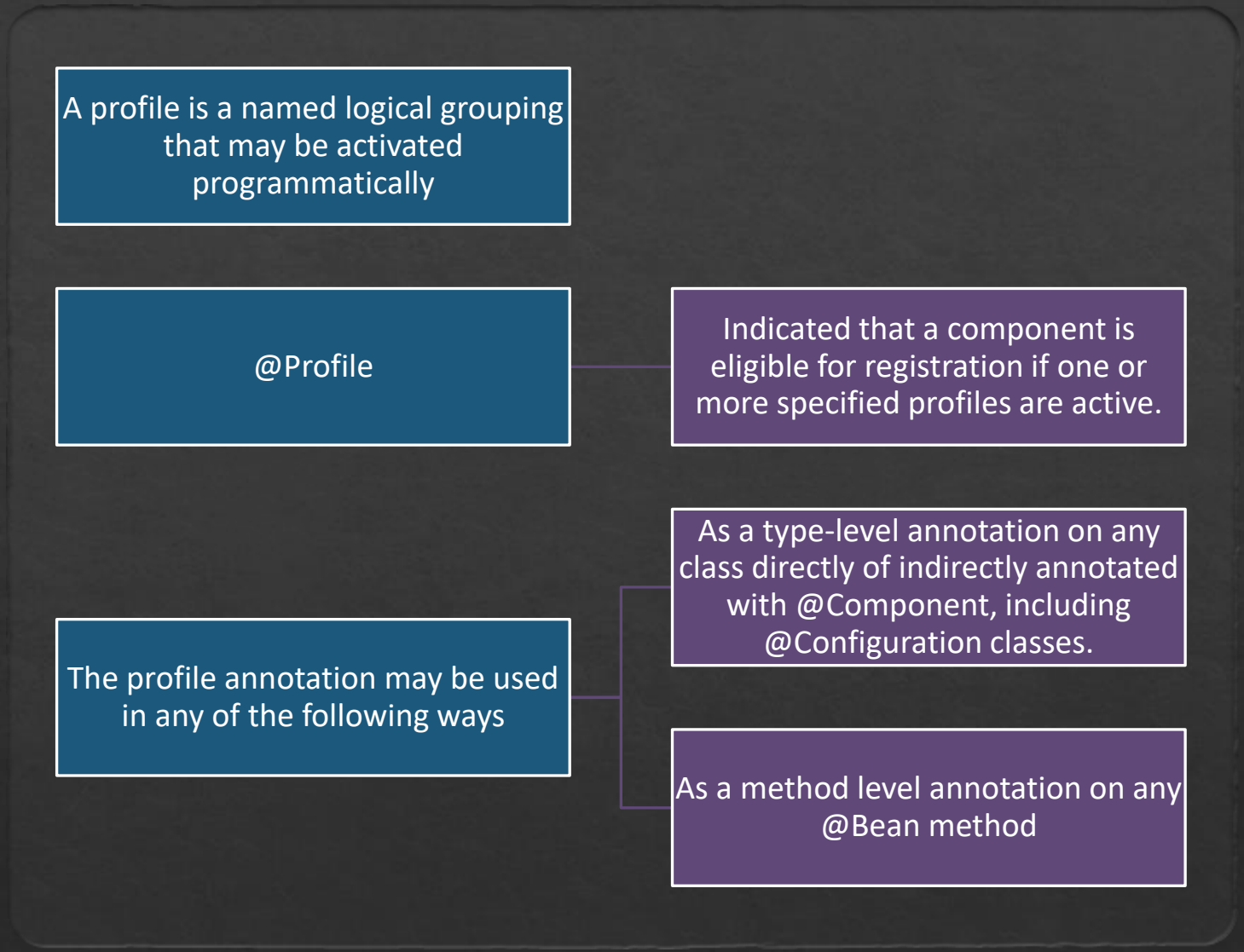
- Marks a constructor, field or setter method as to be autowired by Spring's dependency injection facilities.
- Only one constructor (at max) of any given bean class may carry this
- annotation.
- Resolves the dependency byType else byName

@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

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

Profiles



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

Bean Scopes

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

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

Lifecycle Methods

3 Mechanisms

- Implement interfaces InitializingBean and DisposableBean
- Provide custom initialization and cleanup methods
- Use the annotations @PostContruct and @PreDestroy

Order of Invocation

- Constructor
- Property Injection
- Initialization methods
- Destroy methods(based on the scope)

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

Aspect Oriented Programming(AOP)

- Aspects enable the modularization of concerns like
  - Transaction management
  - Security
  - Logging
- Such concerns are termed crosscutting concerns in AOP literature

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

AOP Concepts

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

- 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

AOP Concepts

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

- 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

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

Spring JDBC

- A value addition provided by Spring as an abstraction over JDBC
- Spring JDBC takes care of the low-level details that makes JDBC a tedious API to develop with.
- Advantages
  - Opens the connection
  - Prepares and executes the statement
  - Handles the code iteration
  - Processes the exceptions
  - Handles transactions
  - Closes the connection, statement and resultset

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

Spring JDBC API

JdbcTemplate

- The classic Spring JDBC class

NamedParameterJdbcTemplate

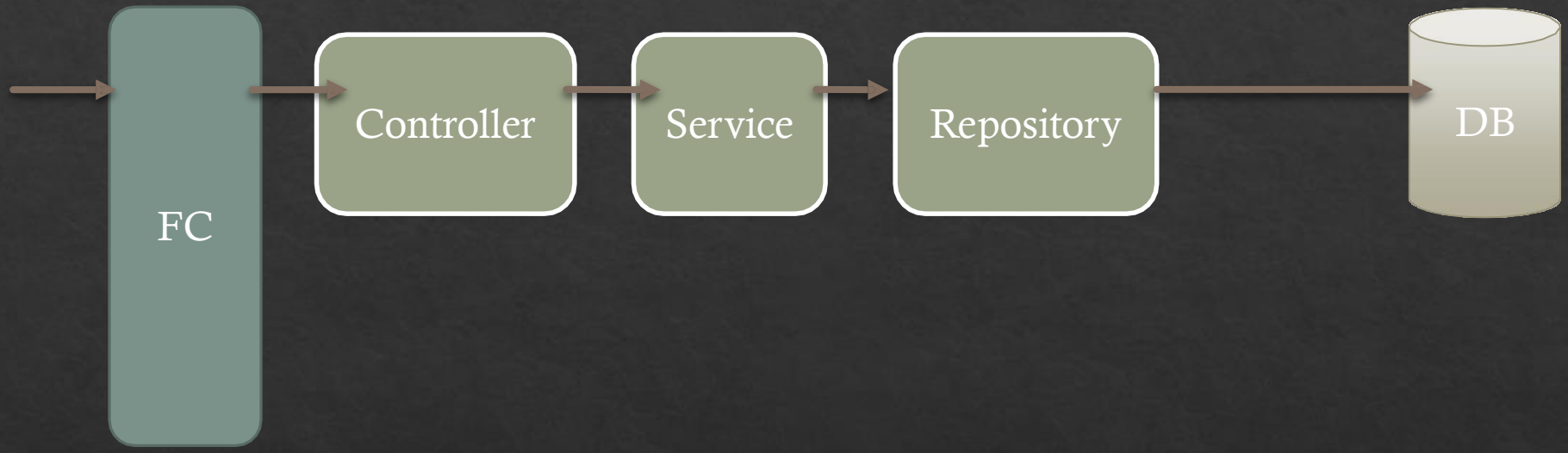
- To provide named parameters instead of the traditional JDBC positional(?) parameters.

SimpleJdbcInsert and SimpleJdbcCall

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

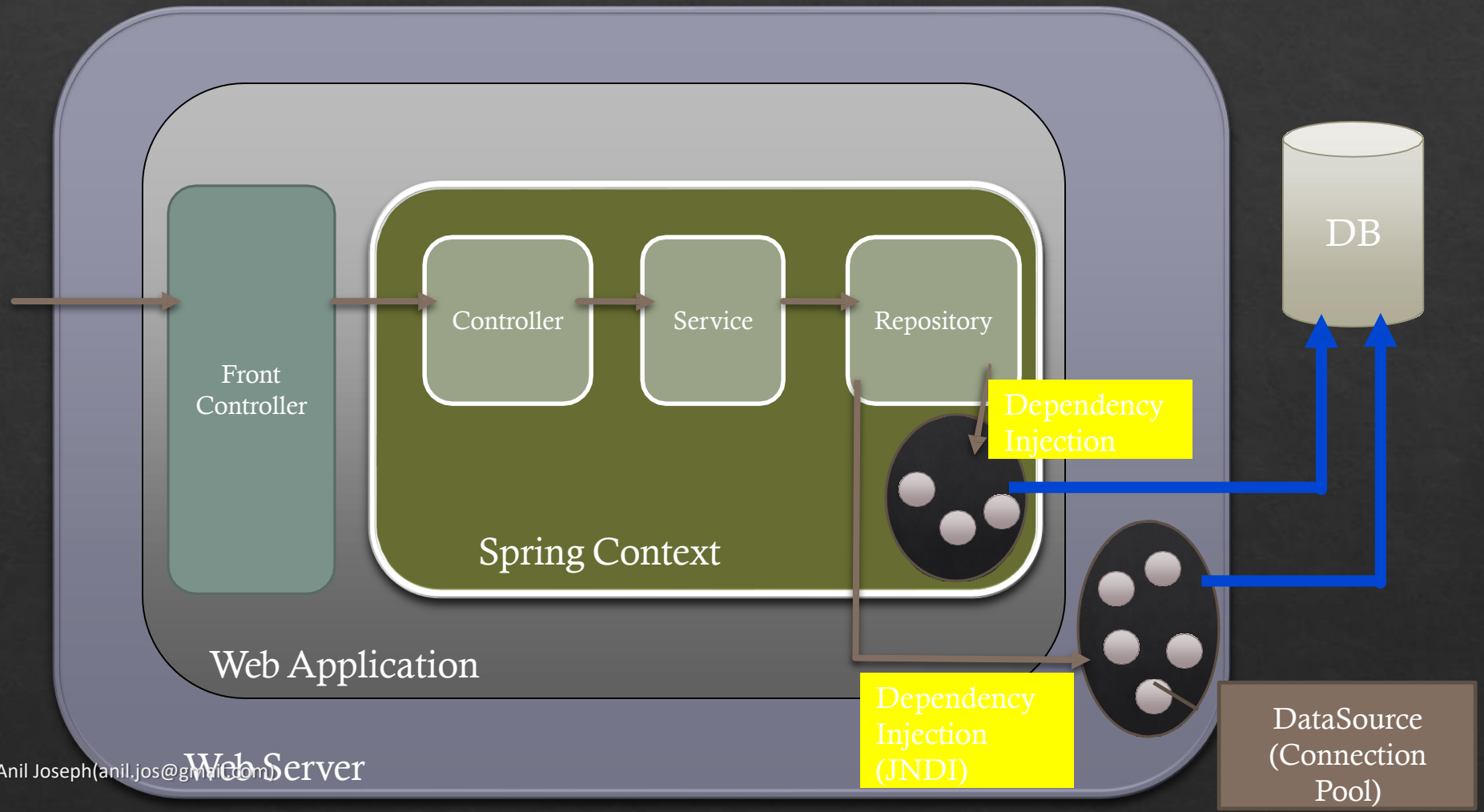
Presented By Anil Joseph(anil.jos@gmail.com)

JDBC Application Design

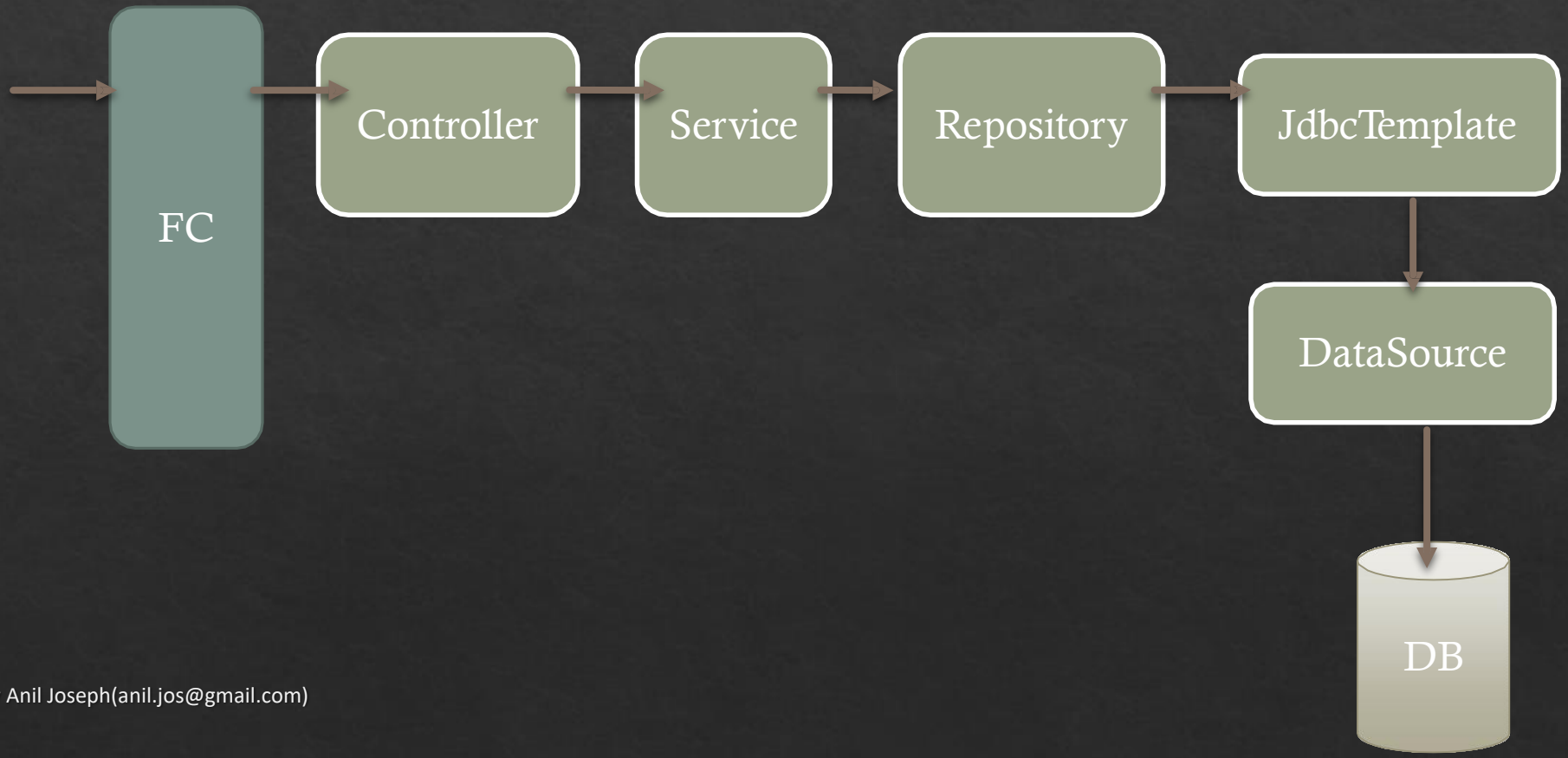


Presented By Anil Joseph(anil.jos@gmail.com)

JDBC Application Design



JDBC Application Design



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

Apache Maven

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

- Maven is a build automation tool used primarily for Java projects.
- Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages.
- Maven defines how to build a project and to manage the dependencies.

Maven pom.xml

- The pom.xml file is the core of a project's configuration in Maven.
- Single configuration file that contains all the information about the project.
- Configure
  - The project information
  - Dependencies
  - Plugins

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

Archetypes

- Archetype is a Maven project templating toolkit.
- Using archetypes provides a great way to enable developers quickly in a way consistent with best practices employed by your project or organization.

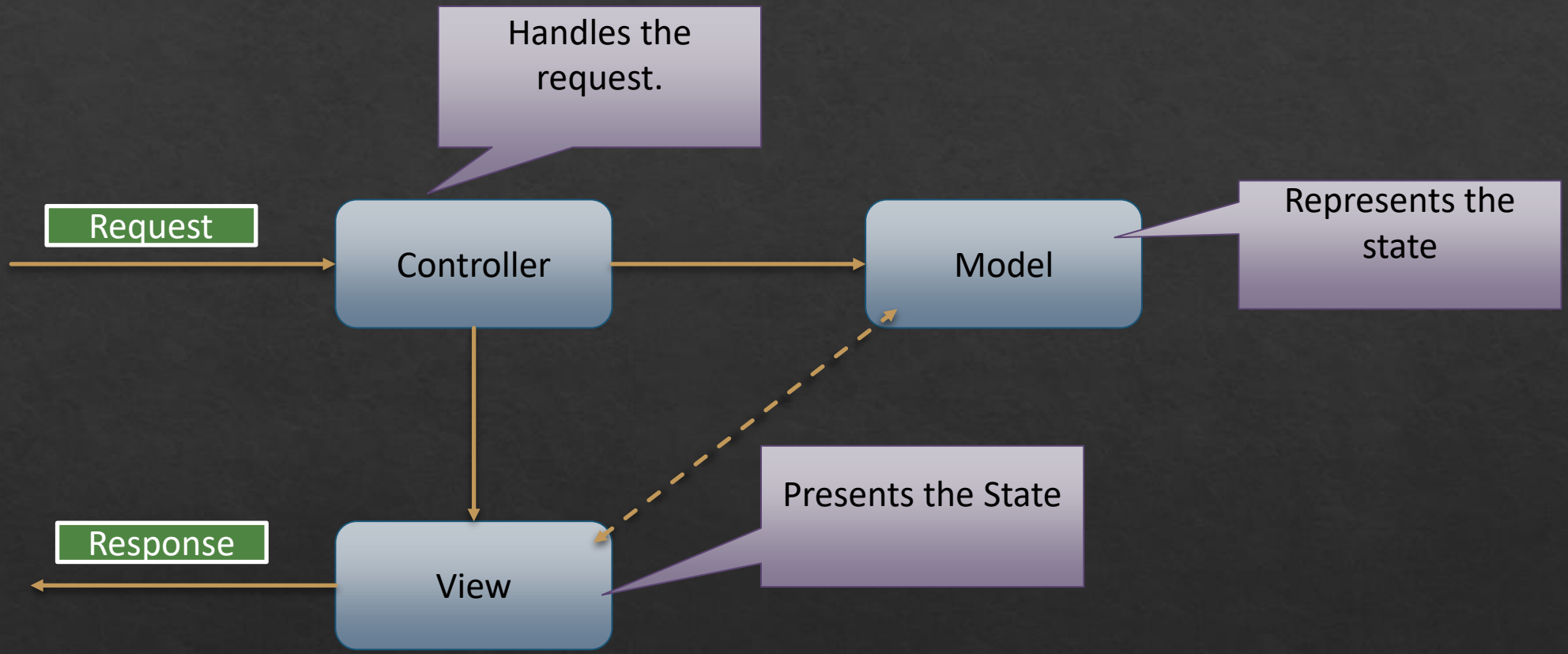
Presented By Anil Joseph(anil.jos@gmail.com)

Spring MVC

- Spring MVC provides a framework to build web applications using the MVC design pattern
- Spring MVC is based in the Servlet specification.

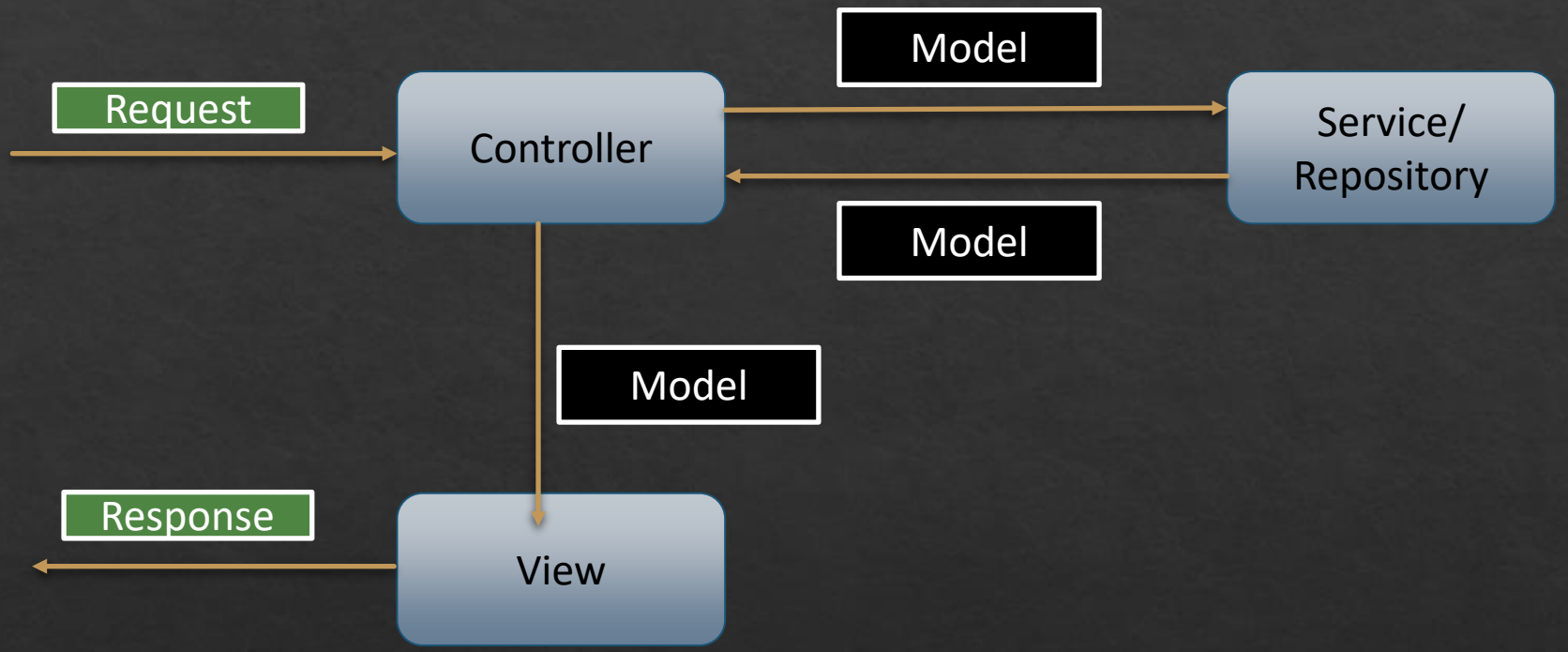
Presented By Anil Joseph(anil.jos@gmail.com)

MVC



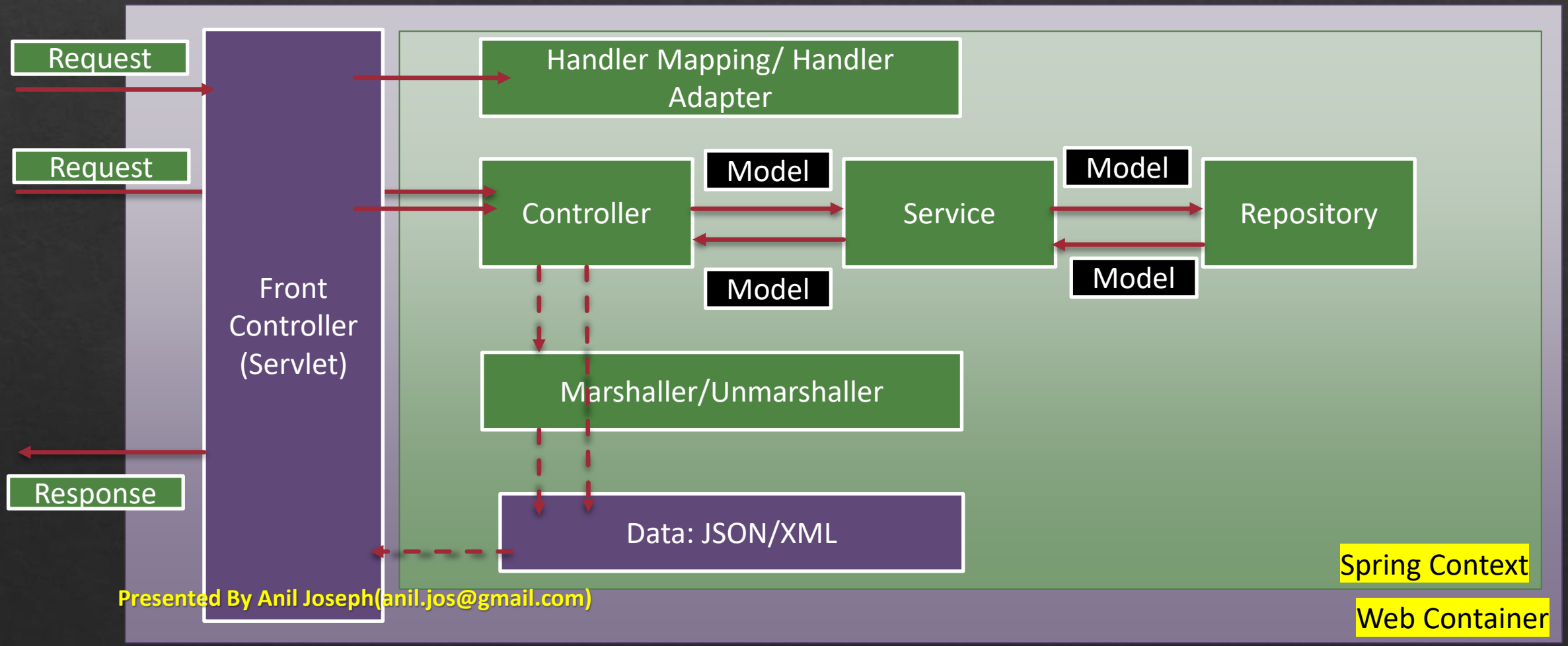
Presented By Anil Joseph(anil.jos@gmail.com)

MVC



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

Spring MVC Stack(REST)



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

Front Controller

The DispatcherServlet is the front Controller in the MVC stack of Spring.

The primary role of the servlet is to dispatch the requests to the controllers for processing

The servlet loads the spring application context and hence has access to all the spring beans and features

The implementing class is ***org.springframework.web.servlet.DispatcherServlet***

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

Controller

Controller is a Spring bean annotated with @Controller annotation.

Use the @RequestMapping annotation to map the requests to the controller

Defines the methods to handle request.

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

Request Handler Methods

Methods in the controller that handle the request can have flexible signatures.

The must be annotated with @RequestMapping to map the request to the handler methods.

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

Request Handler Method Arguments

- Request and Response Object
- Session Object
- InputStream or Reader
- OutputStream or Writer
- Annotated method parameters

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

MVC Annotations

@RequestParam	Maps a request(query) parameter to the method argument
@PathVariable	Maps a path on the request to the method argument
@CookieValue	Maps the value of a request cookie to the method argument
@RequestHeader	Maps a request header value to the method argument
@ModelAttribute	Maps request(query) parameters to a model object

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

Request Handler Return Types

- A string value that is interpreted as the logical view name or as the redirect URL
- An instance of ModelAndView
- An instance of View
- A String values that is the response
  - Use the @ResponseBody annotation

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

REST API(Services)

What is REST?

- Representational State Transfer (REST) is a style of architecture.
  - Describes how networked resources are defined and addressed.
- These principles were first described in 2000 by Roy Fielding
- REST has proved to be a popular choice for implementing Web Services.

Principles of REST

- Client-Server
- Cacheable
- Stateless
- Uniform Interface
- Layered
- Code on demand

Principles of REST

Client–server

- The client and the server both have a different set of concerns.
- The server stores and/or manipulates information and makes it available to the user in an efficient manner.
- The client takes that information and displays it to the user and/or uses it to perform subsequent requests for information.

Stateless

- A communication between the client and the server always contains all the information needed to perform the request.
- There is no session state in the server, it is kept entirely on the client's side.

Cacheable

- The client, the server and any intermediary components can all cache resources in order to improve performance.

Principles of REST

Uniform Interface

- Provides a **uniform interface** between components.
- Requests from different clients look the same

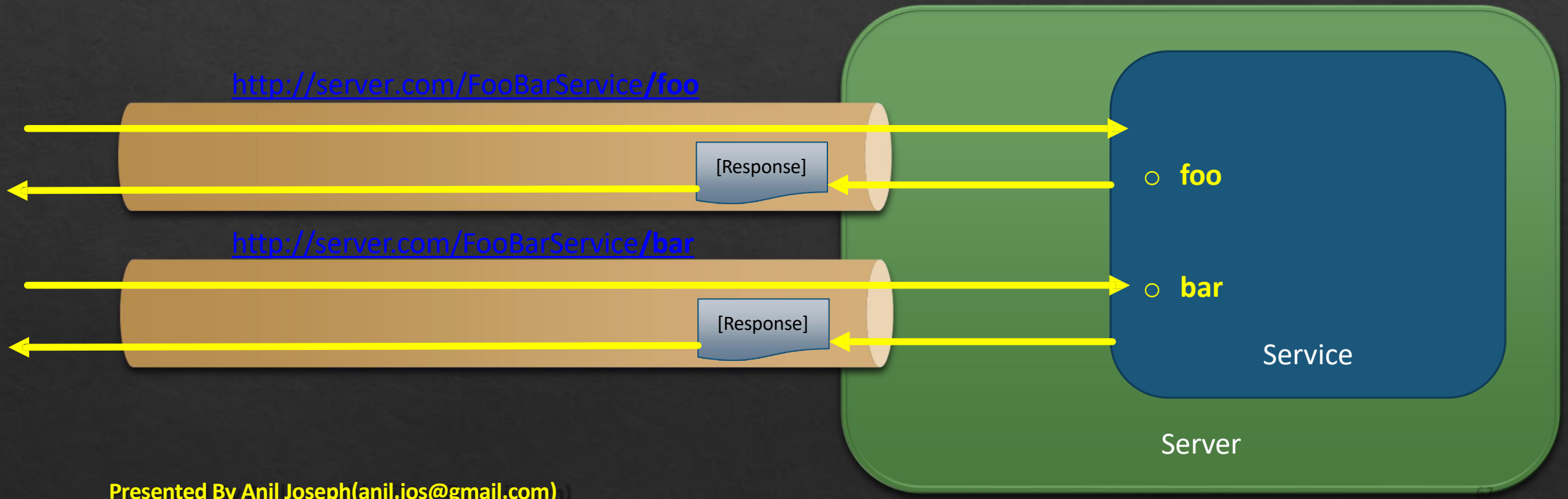
Layered System

- Between the client requests and the server response there can be a number of components/servers in the middle.
- The layers can provide security, caching, load-balancing or other functionality.
- The client is agnostic as to how many layers.

Code on demand

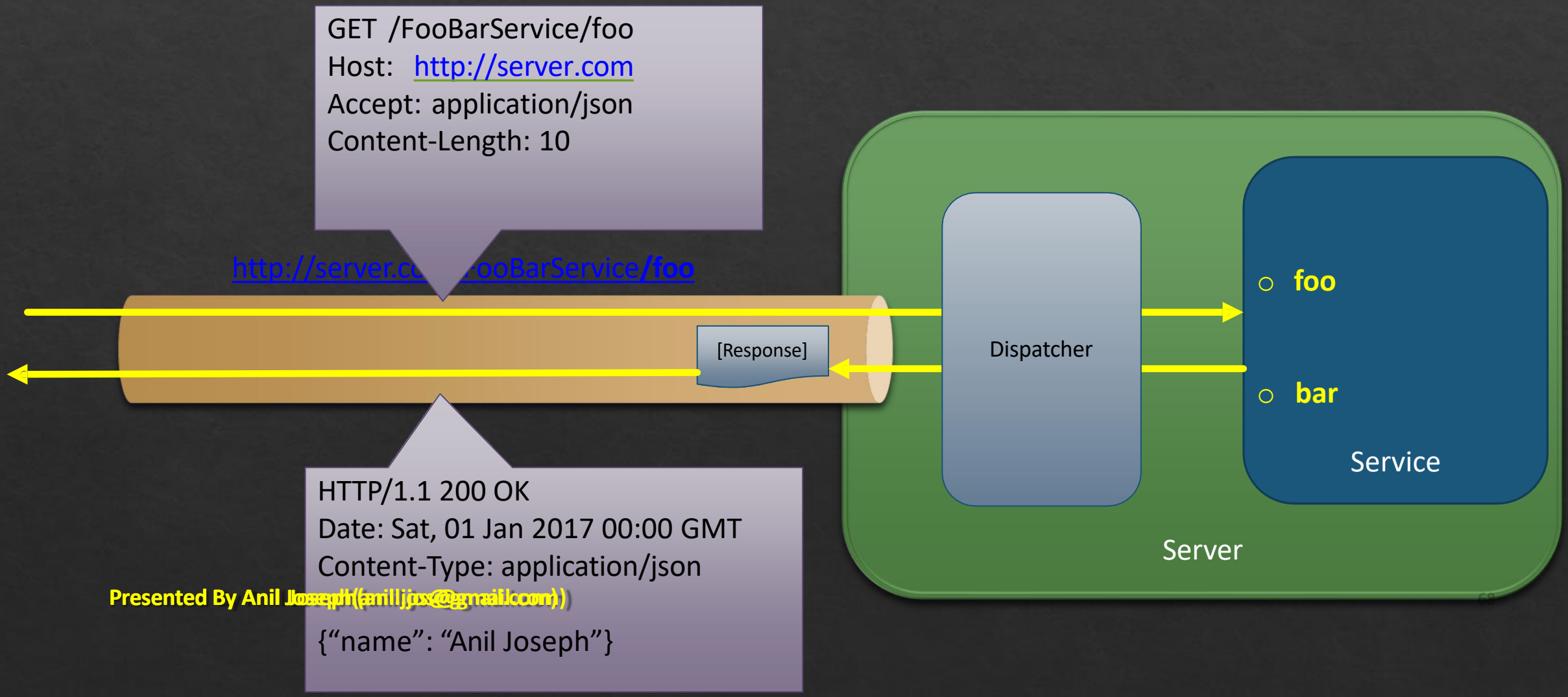
- This constraint is optional
- The client can request code from the server, and then the response from the server will contain some code

REST Service



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

REST Service



Http Methods

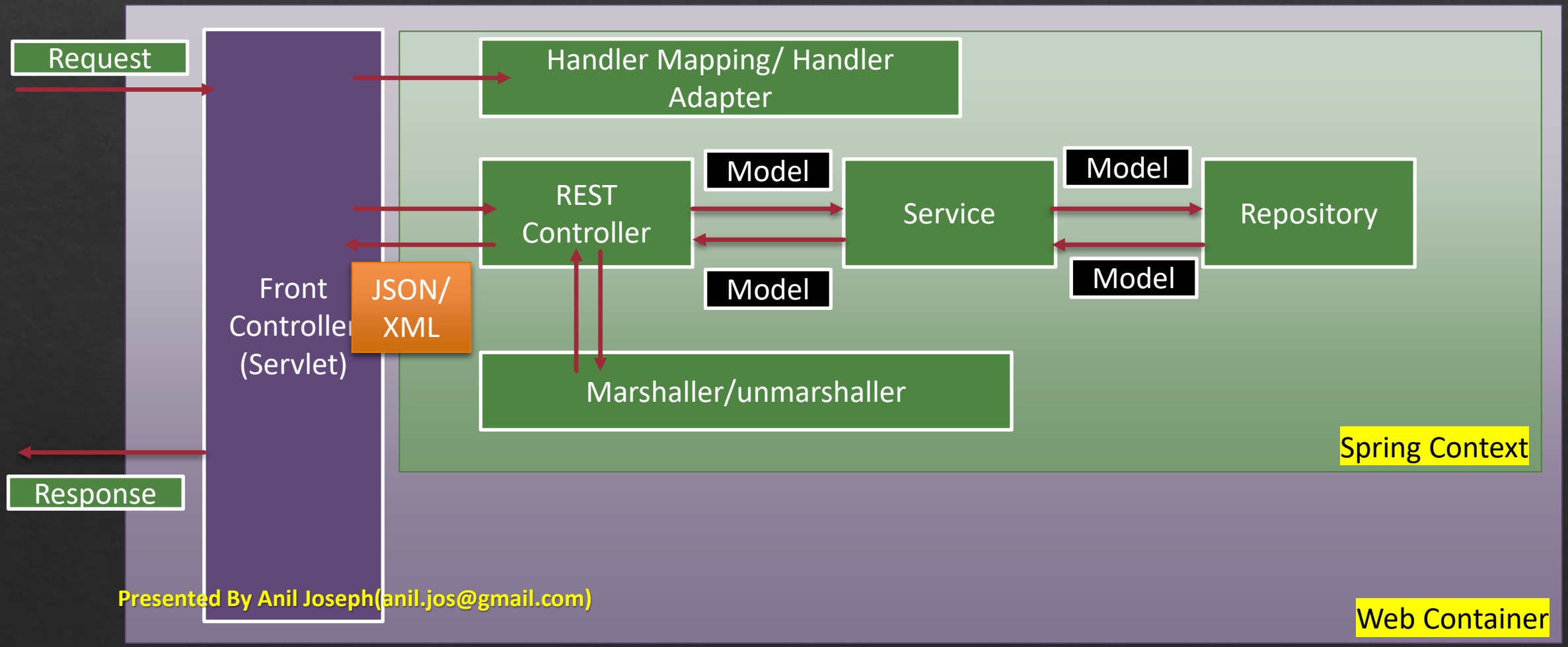
Get	To retrieve or read a resource.
Post	To create a new resource.
Delete	To delete/remove an existing resource.
Put	To update an existing resource.
Patch	Used to update a slice of the resource

Spring MVC REST

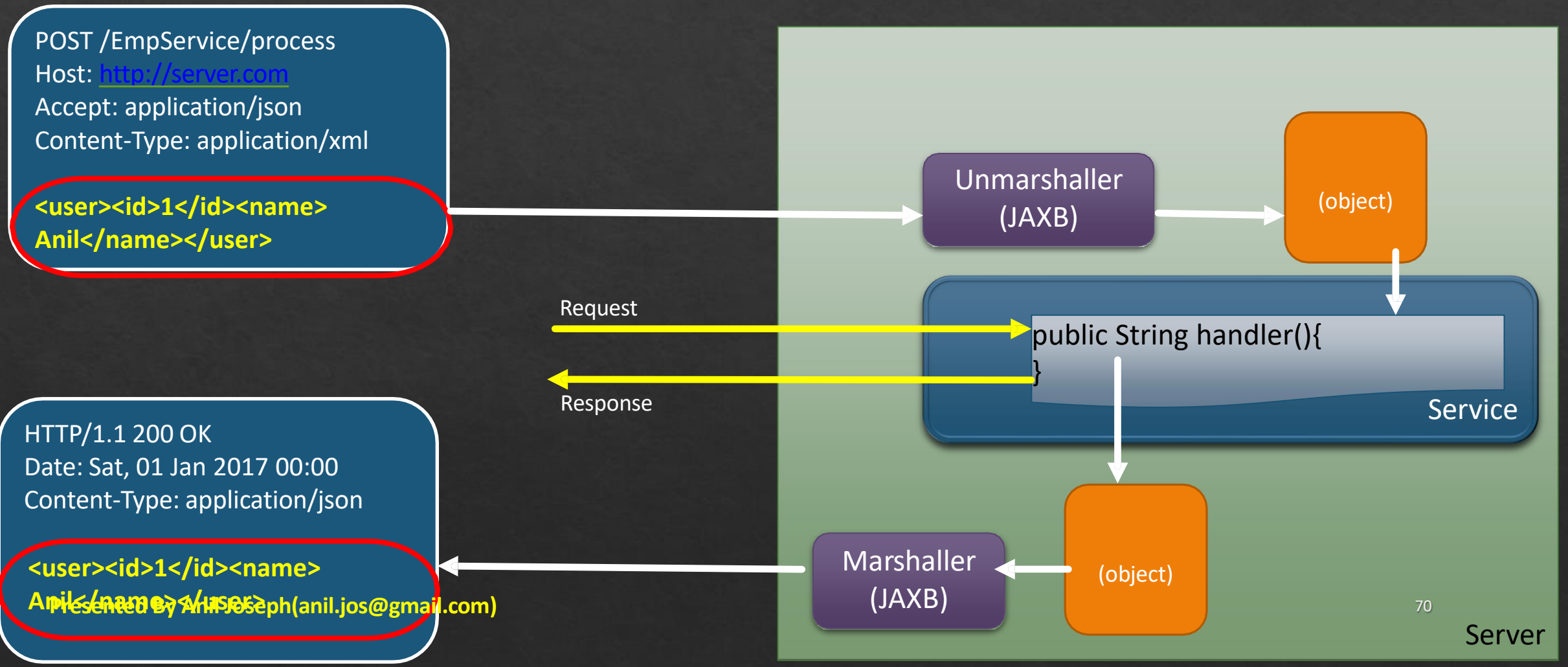
- Spring MVC supports the creation of REST Services
- @RestController introduced in Spring 4.0
  - Defines a controller for REST
  - Combines @Controller and @ResponseBody
- @RequestMapping
  - Maps a method to a request URL

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

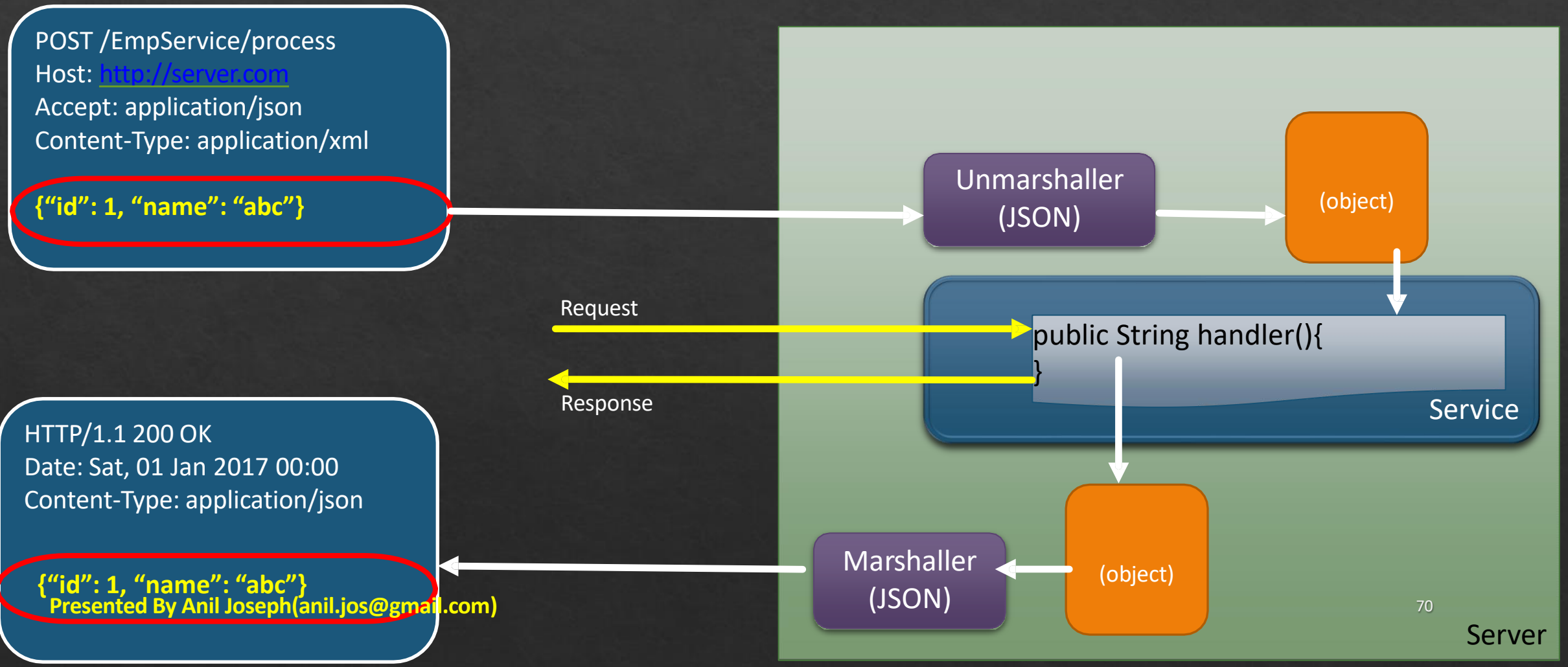
Spring MVC Stack



XML Marshalling and Unmarshalling



JSON Marshalling & Unmarshalling



ORM

- ORM stands for Object-Relational Mapping.
- It is a technique for converting data between incompatible type systems (objects in Java and relational databases).
- Provides a way to interact with the database using Java objects.
- Common ORM frameworks: Hibernate, JPA (Java Persistence API), EclipseLink.

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

Benefits of Using ORM

- Simplifies database interactions.
- Reduces boilerplate code.
- Supports complex queries with ease.
- Provides a consistent API for database operations.
- Facilitates database portability.

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

Spring Data JPA

- Spring Data JPA is a part of the Spring Framework.
- Simplifies the implementation of JPA-based repositories.
- Reduces the amount of boilerplate code needed for data access layers.
- Integrates seamlessly with Spring Boot for quick setup.

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

Mapping an Entity class

- @Entity
  - Marks the class as a JPA entity.
- @Table
  - Specifies the table name in the database.
  - Example: @Table(name = "users")
- @Id
  - Denotes the primary key of the entity.
- @GeneratedValue
  - Specifies the strategy for primary key generation.
  - Example: @GeneratedValue(strategy = GenerationType.IDENTITY)

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

Mapping an Entity class

- @Column
  - Maps the field to a column in the database.
  - Can specify column name, nullable, unique, etc.
  - Example: @Column(name = "email", nullable = false, unique = true)
- @OneToOne
  - Defines a one-to-one relationship between entities.
  - Example: @OneToOne(cascade = CascadeType.ALL)
- @OneToMany
  - Defines a one-to-many relationship.
  - Example: @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
- @ManyToOne
  - Defines a many-to-one relationship.
  - Example: @ManyToOne(fetch = FetchType.LAZY)

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

Mapping an Entity class

- @ManyToMany
  - Defines a many-to-many relationship.
  - Example: @ManyToMany
- @JoinColumn
  - Specifies the foreign key column.
  - Example: @JoinColumn(name = "user_id")
- @JoinTable
  - Specifies the join table for many-to-many relationships.
- @Embedded
  - Embeds a component or value object.
- @Embeddable
  - Marks a class as embeddable within an entity.

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

JPA Key interfaces

- CrudRepository<T, ID>: Provides basic CRUD operations.
- PagingAndSortingRepository<T, ID>: Adds methods for pagination and sorting.
- JpaRepository<T, ID>: Extends CrudRepository and PagingAndSortingRepository, providing additional methods for JPA-specific operations.
- Automatic Implementation:
  - Spring Data JPA dynamically generates the implementation for the repository interface at runtime, based on the methods defined in the interface.
  - This means you don't need to write the actual implementation code.

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

Defining a Repository

- Extend JpaRepository to create repository interfaces.
- Example

```
public interface UserRepository extends JpaRepository<User, Long> {
    List<User> findByName(String name);
}
```

- Spring Data JPA provides CRUD operations out-of-the-box.
- Define query methods by following naming conventions.
- Example: findByEmail, findByAgeGreaterThan.
- Supports complex queries using keywords (And, Or, Between, etc.).

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

Spring Transactions

Provides declarative transaction management.

Seamless support for both native transactions(local) and J2EE server transactions (JTA, global)

Flexible replacement for EJB CMT

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

Configure Spring Transactions

Define the Transaction Manager

Define the Transaction Propagation(boundaries)

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

Transaction Managers

Native transaction managers

- JDBC: DataSourceTransactionManager
- Hibernate: HibernateTransactionManager
- JPA: JPATransactionManager

Native strategies work in any environment

- Only against a single database
- No distributed transaction coordination
- leverage full power of underlying resource
- Isolation levels, savepoints, etc

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

Transaction Managers

- JTA-based transaction coordinator
  - Spring's JTATransactionManager adapter
- Java Transaction API to talk to an external transaction coordinator
  - Typically using the standard XA protocol to coordinate heterogeneous transactional resources

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

@Transactional

- Use @Transactional to wrap a method in a database transaction.
- Used to set
  - Propagation
  - Isolation Level
  - Timeout
  - Read-only
  - Rollback conditions.
- Used to specify the transaction manager.

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

Transaction Propagations

- REQUIRED
  - The default propagation.
  - Spring checks if there is an active transaction, if none exist then it creates a new one
  - Otherwise, the method appends to the currently active transaction.
- SUPPORTS
  - Spring first checks if an active transaction exists.
  - If a transaction exists, then the existing transaction will be used.
  - If there isn't a transaction, the method is executed non-transactional

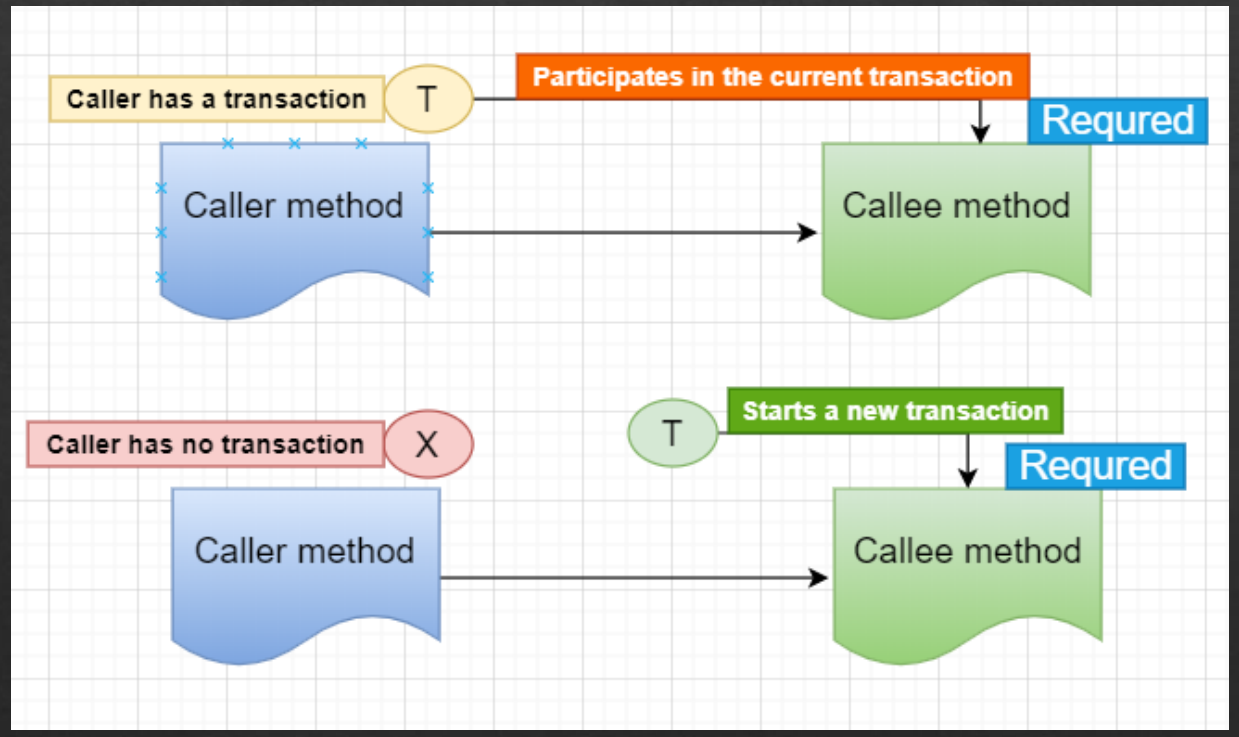
Presented By Anil Joseph(anil.jos@gmail.com)

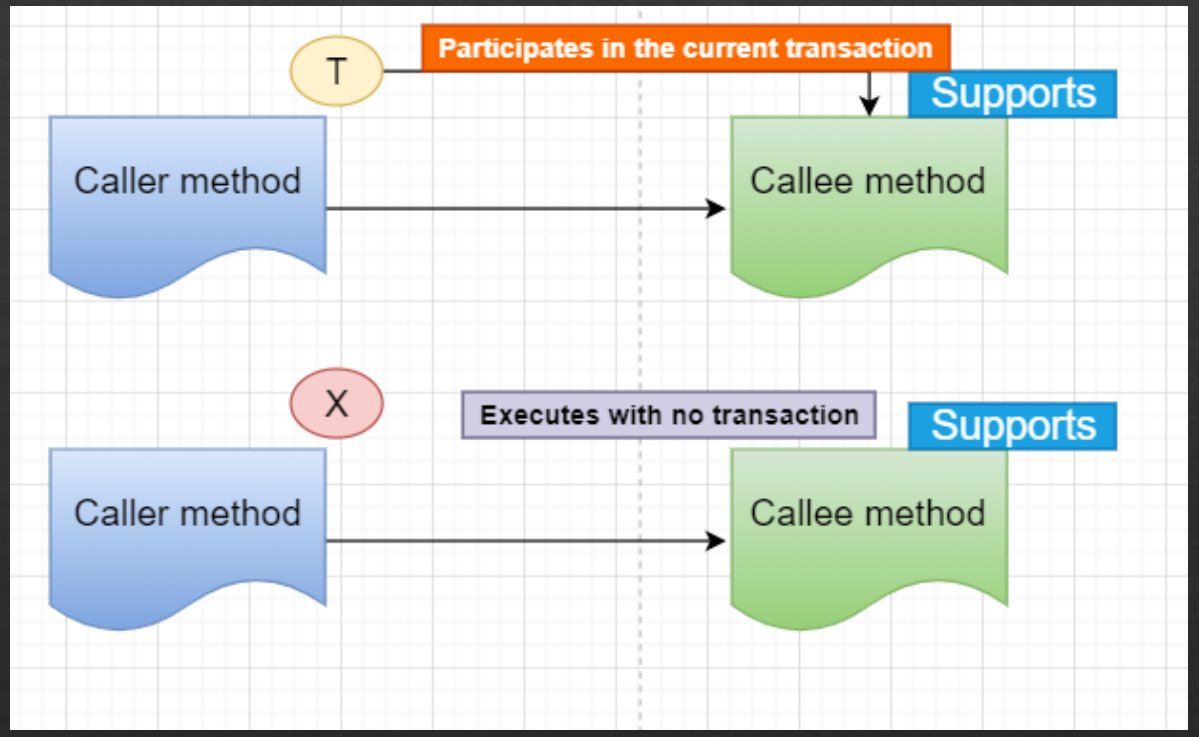
Transaction Propagations

- NOT_SUPPORTED
  - Spring at first suspends the current transaction(if one exists) if it exists and the method executed without a transaction.
- REQUIRES_NEW
  - Spring suspends the current transaction if it exists and then creates a new one.
- MANDATORY
  - If there is an active transaction, then it will be used.
  - If there isn't an active transaction, then Spring throws an exception
- NEVER
  - Spring throws an exception if there's an active transaction:

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

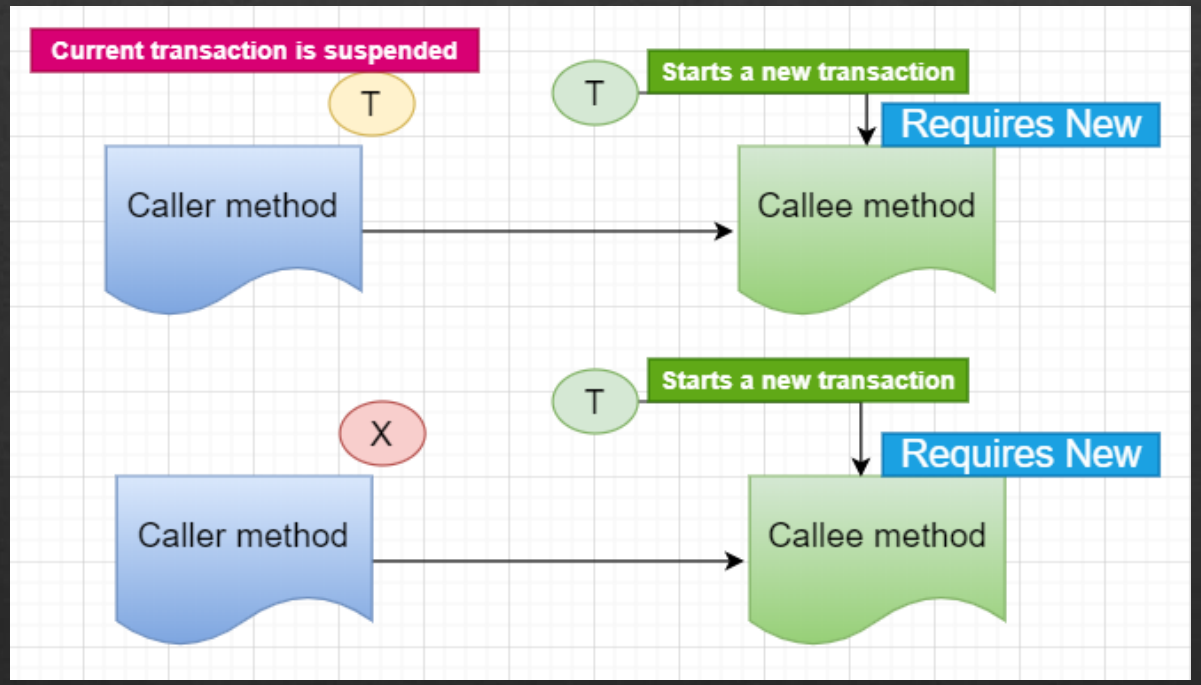
Propagation Behaviors

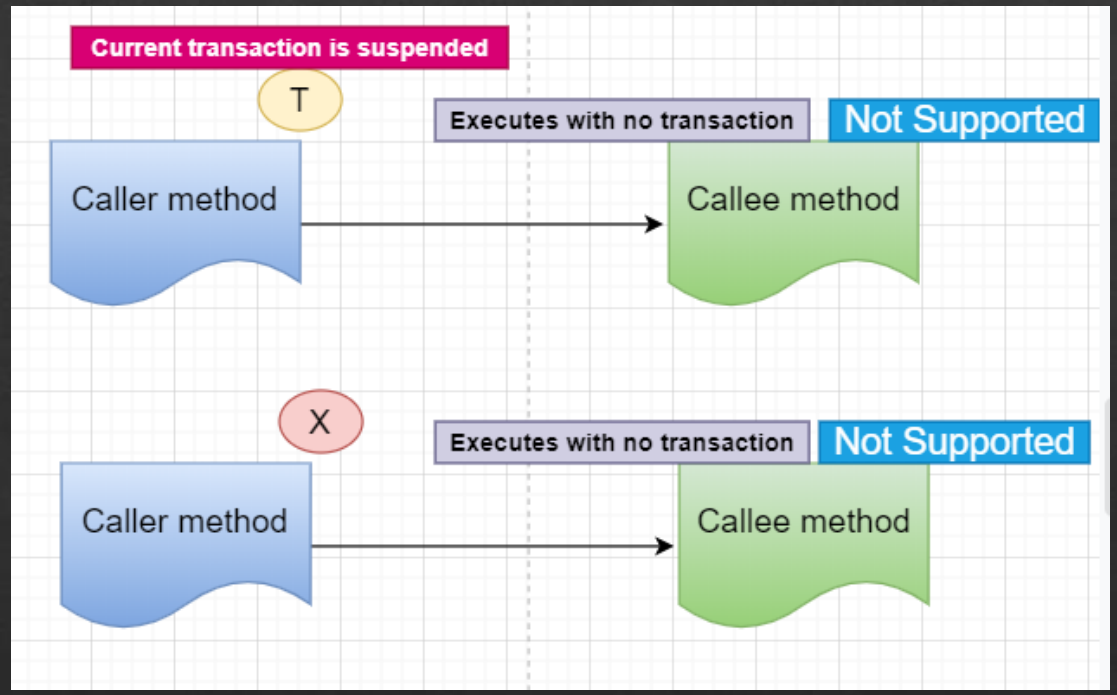




Presented By Anil Joseph(anil.jos@gmail.com)

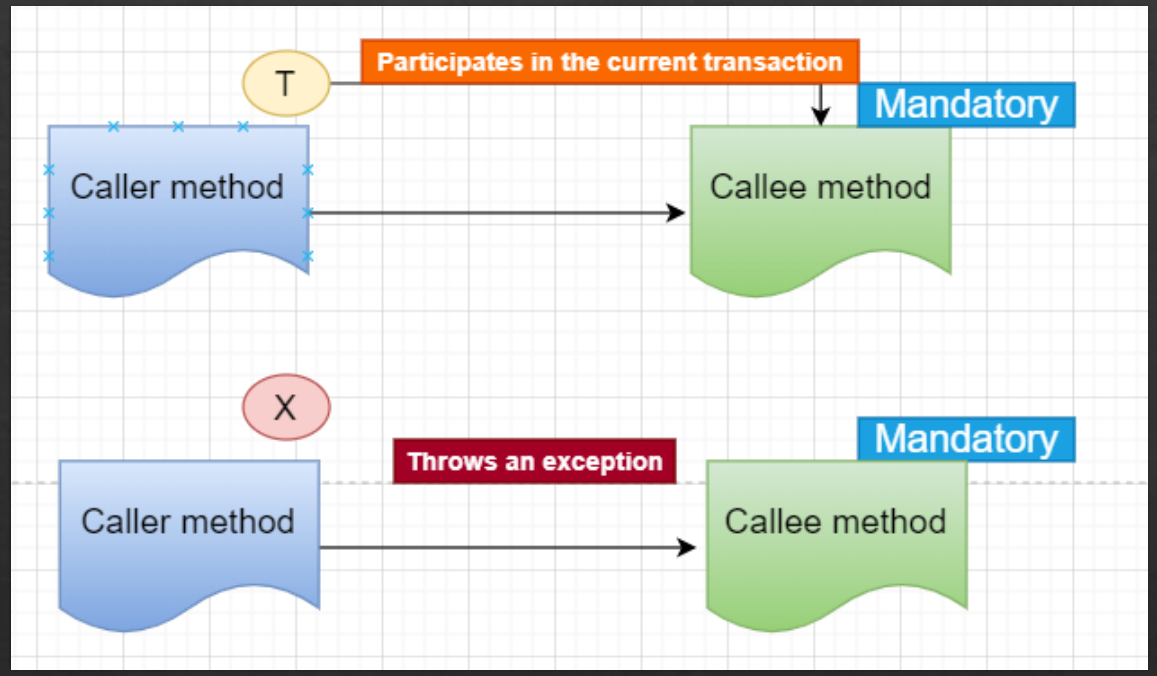
Propagation Behaviors

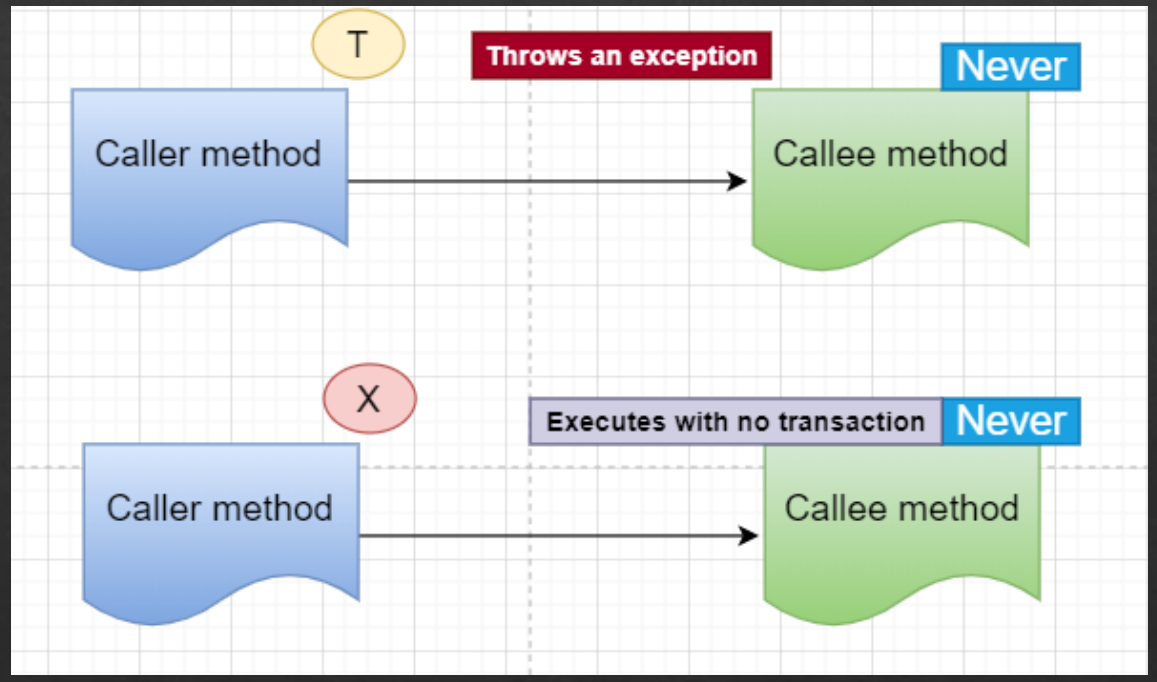




Presented By Anil Joseph(anil.jos@gmail.com)

Propagation Behaviors





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

Actuator

- Spring Actuator is a powerful set of tools for monitoring and managing Spring Boot applications.
- It provides various production-ready features like health checks, metrics, application information, and more.

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

Application Monitoring

- Health Checks:
  - Actuator provides built-in health checks that help ensure your application is running correctly.
  - It checks the status of components like databases, message brokers, and custom health indicators, allowing you to quickly identify and address issues.
- Metrics:
  - Actuator exposes various metrics related to the application's performance, such as memory usage, garbage collection, thread pool statistics, and HTTP request metrics.
  - These metrics are invaluable for monitoring and optimizing the performance of your application.

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

Operational Insights

- Application Information:
  - Actuator endpoints provide detailed information about the application's build, environment properties, and configuration.
  - This helps in understanding the current state of the application and diagnosing issues more effectively.
- Tracing and Auditing:
  - Actuator supports tracing and auditing of HTTP requests, which can be crucial for identifying bottlenecks, understanding request flows, and ensuring security compliance.

Troubleshooting

- Detailed Error Reports:
  - When things go wrong, Actuator can provide detailed error reports and stack traces, making it easier to debug issues.
- Loggers Endpoint:
  - The /actuator/loggers endpoint allows you to view and modify the logging levels of your application at runtime.
  - This is particularly useful for debugging in production environments without restarting the application.

Thank You

(anil.jos@gmail.com)