Spring Boot

Charles Moloney

Advanced Java Learning Workshops: Week 4

Agenda

1. What is a Framework?
2. Introduction to Spring
3. MVC Code Demo
4. Questions

What is a Framework?

Framework

- **You give the framework control and framework calls your code**
- More comprehensive “paradigm shift”
- Inversion of Control: Framework manages the flow and calls your code via callbacks/delegations

Library

- **You keep control and call the libraries**
- Collection of pre-written code you call explicitly
- Can be easily added/removed from existing projects

Advantages of Frameworks

- Productivity: Less boilerplate, conventions over configuration.
- Maintainability: Standardized project structure, patterns.
- Testability: Integrated support for mocking, dependency injection simplifies testing.
- Scalability & Extensibility: Plug-ins and modules reduce reinvention
- Security: Built in compliance and security features

Faster development, fewer bugs*

Introduction to Spring

What is it?

- Comprehensive Java Framework with many modules assisting with handling java objects.
  - Modular Design
- Promotes loose coupling and dependency injection
  - Instead of hardcoding dependencies, Spring injects them for you
- IoC-based: Instantiates, configures, and assembles Beans.
  - Beans: Class with noarg constructor (usually w/ getters and setters) that can be created and injected as needed

A bit more on Beans

```java
import org.springframework.stereotype.Component;

@Component
public class Engine {
    public void start() {
        System.out.println("Engine started");
    }
}
```

OR

```java
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfig {

    @Bean
    public Engine engine() {
        return new Engine(); // Engine class defined elsewhere
    }
}
```

```java
@Component
public class CarService {
    @Autowired
    private Engine engine;
}
```

OR

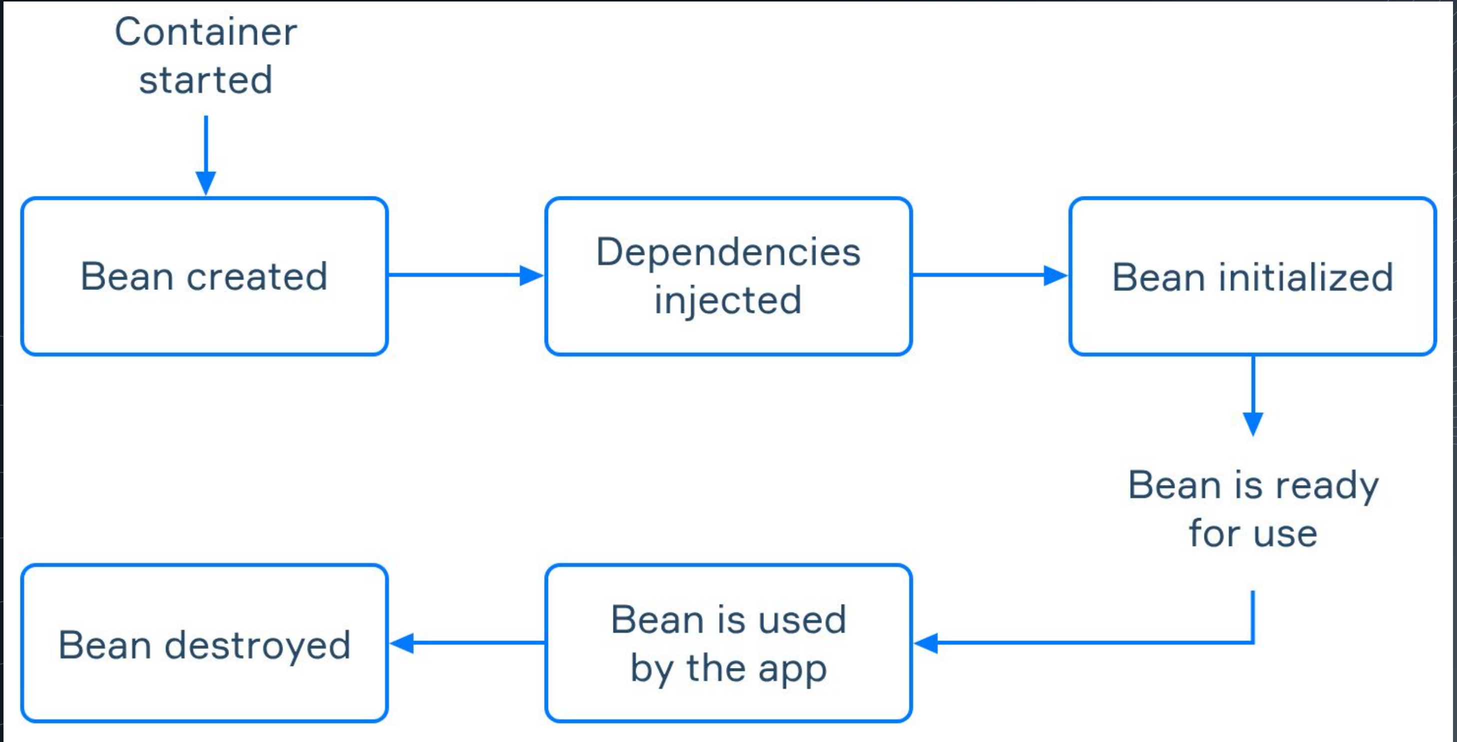
```java
@Component
public class CarService {
    private final Engine engine;

    @Autowired
    public CarService(Engine engine) {
        this.engine = engine;
    }
}
```

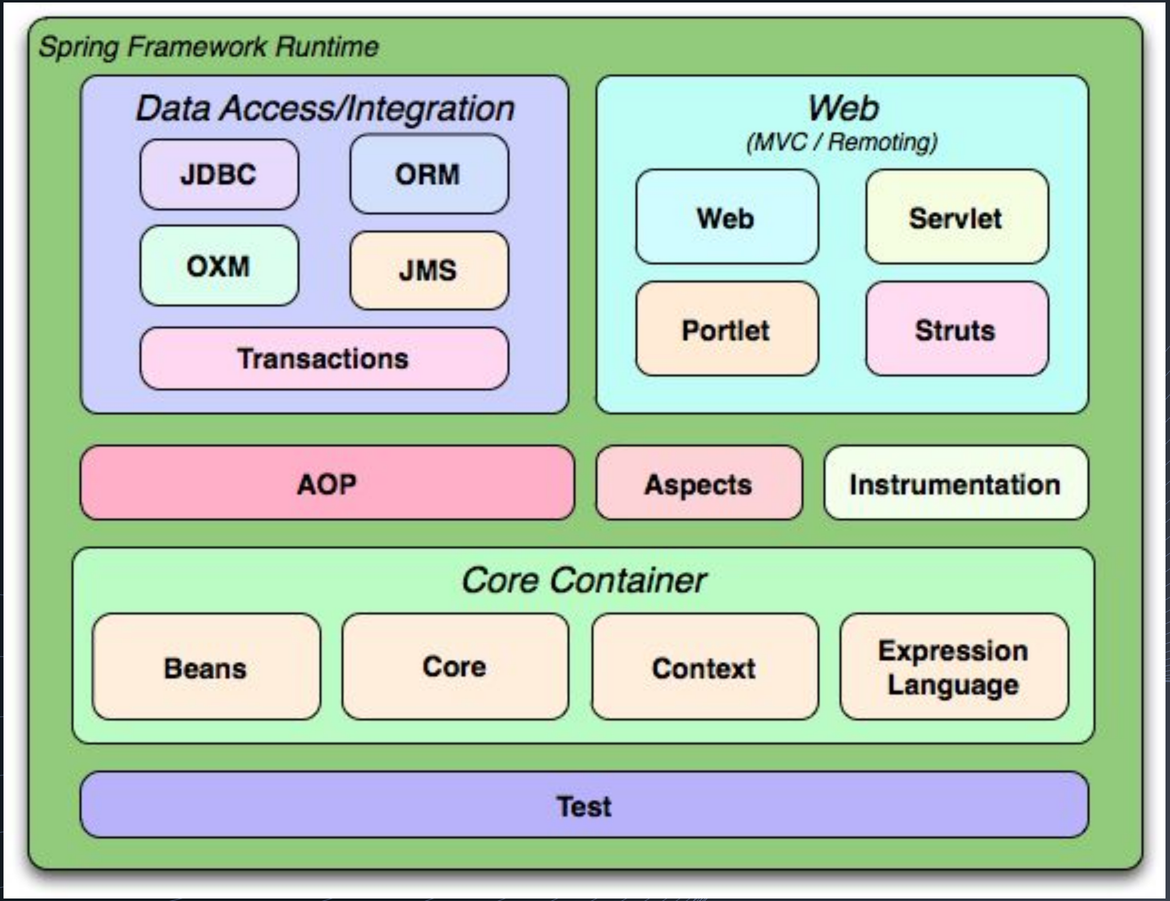
OR

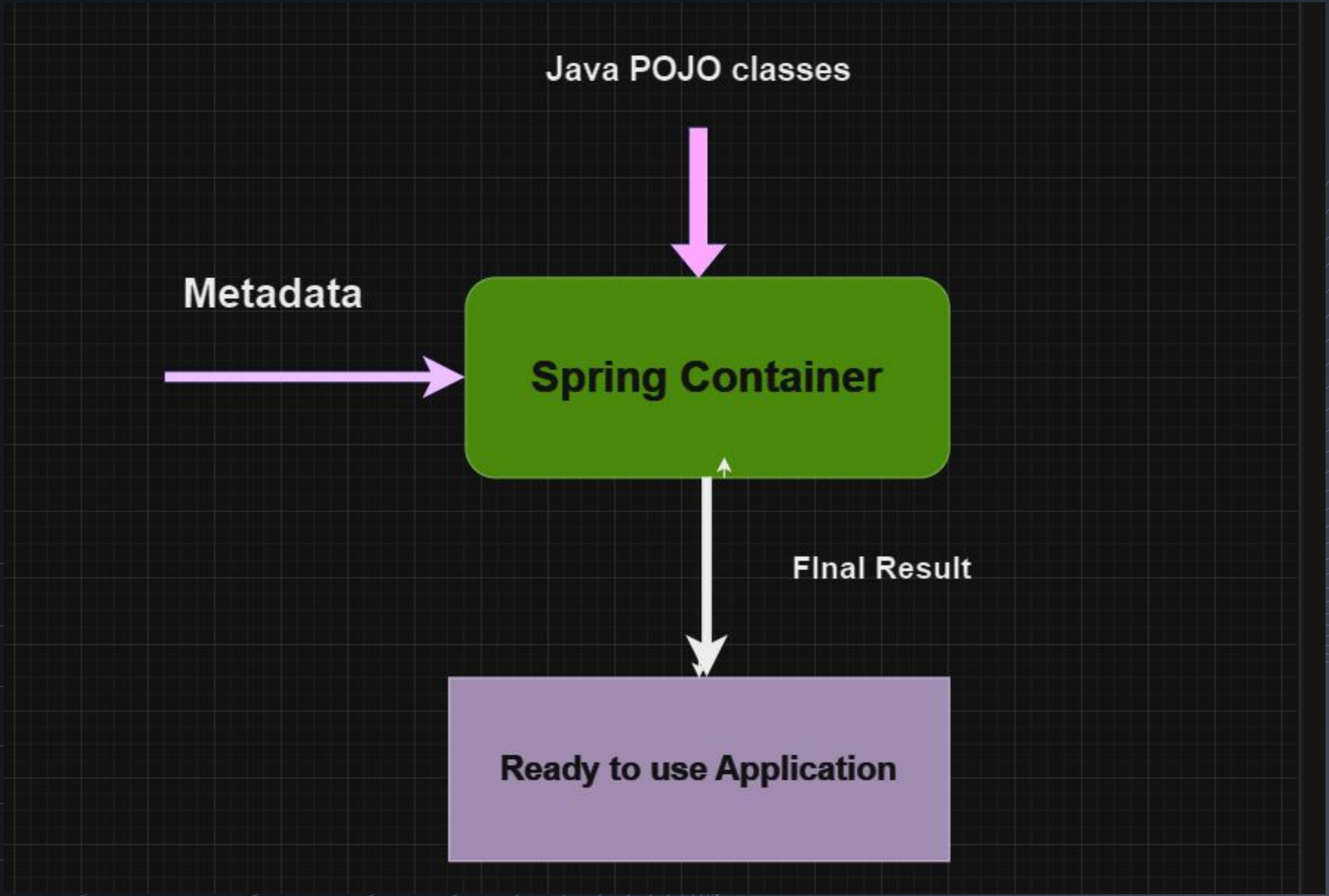
```java
@Component
public class CarService {
    private Engine engine;

    @Autowired
    public void setEngine(Engine engine) {
        this.engine = engine;
    }
}
```

A bit more on Beans

- Spring is aware of the beans throughout your Application Context and can add/remove them as needed through @Autowired




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


Java POJO classes
Metadata
Spring Container
FInal Result
Ready to use Application

Spring Boot

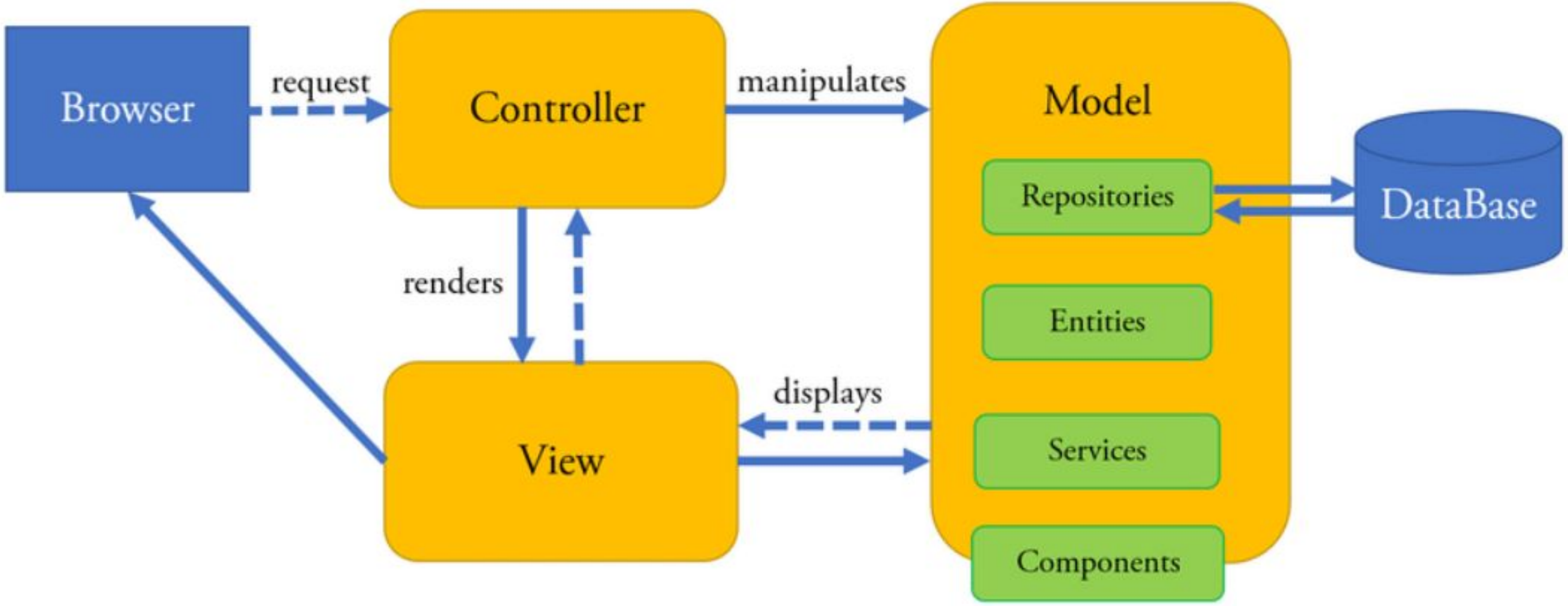
- **Opinionated, heavier version of Spring that removes a lot of boilerplate and needed configuration**
    - This is what we are using today
    - No ApplicationContext, no/less xml, fewer dependencies with spring “starters”
    - For comprehensive list of differences, see https://www.baeldung.com/spring-vs-spring-boot
- Used to create, standalone, production ready apps
- Built in health checks, metrics, etc.
- Spring Initalizr
- “Press one button to start”

```java
@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

MVC Code Demo

Model-View-Controller

- Model: Manages data/services and interfaces with databases
- View: What the user sees and interacts with
- Controller: Intermediary between the Model and View



cURL examples

```
# List all todos
curl -X GET http://localhost:8080/api/todos

# Get a single todo (ID = 1)
curl -X GET http://localhost:8080/api/todos/1

# Create a new todo
curl -X POST http://localhost:8080/api/todos \
  -H "Content-Type: application/json" \
  -d '{"title":"Buy milk"}'

# Fully change a todo
curl -X PUT http://localhost:8080/api/todos/1 \
  -H "Content-Type: application/json" \
  -d '{"title":"Buy chocolate milk","completed":true}'

# Delete a todo (or mark complete via MVC)
curl -X DELETE http://localhost:8080/api/todos/1

# Mark a task complete
curl -X POST http://localhost:8080/api/todos/complete/1
```

Questions?