SPRING ENTERPRISE FRAMEWORK

Appreciating All Levels From Surface to Depth

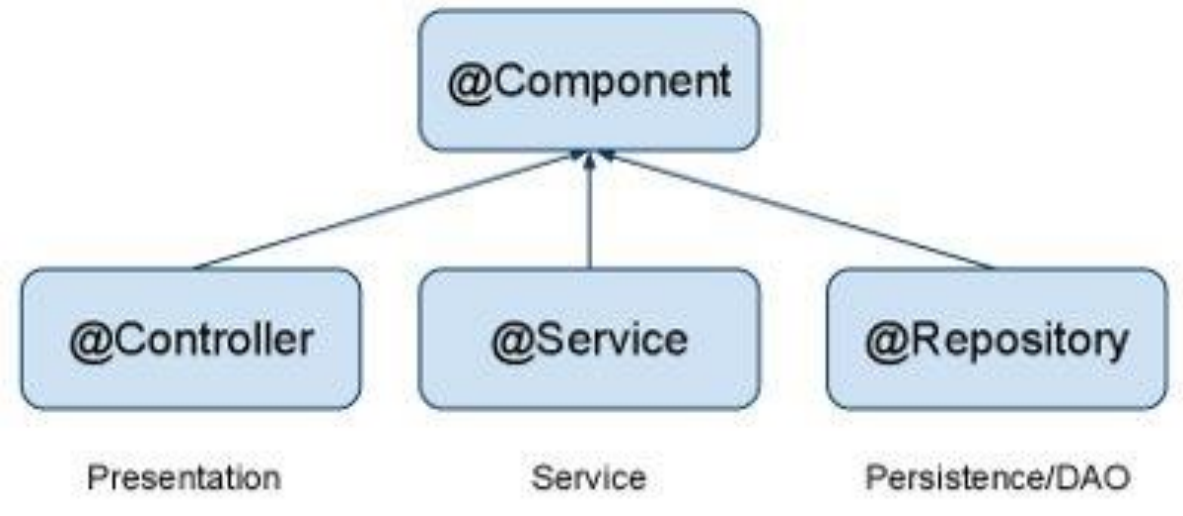

IO EXECUTION
Spring XD
Boot
Grails
IO COORDINATION
Cloud
S
IO FOUNDATION
Integration
Batch
Big Data
Web
Relational
Data Access
DATA
Non-Relational
Data Access
CORE
Framework
Security
Groovy
Reactor

Spring Framework

- Infrastructure support for developing Java applications.
- Configure disparate components into a fully working application ready for use.
- Build applications from “plain old Java objects” (POJOs)
- Non-intrusive - domain logic has little or no dependencies on framework
- Lightweight application model is that of a layered [N-tier] architecture. Spring 3 Tiers:

1. Presentation objects such as Spring MVC controllers are typically configured in a distinct ***presentation context[tier]***
2. Service objects, business-specific objects, etc. exist in a distinct ***business context[tier]***
3. Data access objects exist in a distinct ***persistence context[tier]***

Backend Components



@Component is a generic stereotype for any Spring-managed component. @Repository, @Service, and @Controller are specializations of @Component for more specific use cases, for example, in the persistence, service, and presentation layers, respectively.

Annotate based on Function

- OPTION - annotate all your component classes with @Component

- Using @Repository, @Service, and @Controller is:
  - Better suited for processing by tools
    - @Repository - automatic translation of exceptions
    - @Controller – rich set of framework functionality
    - @Service – “home” of @Transactions
  - More properly suited for associating with aspects
  - May carry additional semantics in future releases of the Spring Framework.

Service Layer

- **Issue: not whether or not it is needed**

BUT

What it contains

Domain Driven Design

- Primary focus - the core domain and domain logic.
- Complex designs based on a model of the domain.
- Collaboration between technical and domain experts to iteratively refine a conceptual model that addresses particular domain problems.
    - GOAL: a Rich Domain Model

“Thin” Domain Model

- Contains objects properly named after the nouns in the domain space
- Objects are connected with the rich relationships and structure that true domain models have.

Extreme case: Anemic Domain Model

Little or no behavior – bags of getters and setters.

Service Layer

- In a perfect world:

"Thin Layer"

With

"Rich Domain Model"

- No business rules or knowledge
- Coordinates tasks
- Delegates work to domain objects

"The Reality"

Quite often additional **"Domain"** Services exist - populated with "externalized" Business/Logic rules.

SERVICE COMPONENT USE CASES

See Service Component use cases.docx

Main Point

- An N Tier Architecture separates an application into layers thereby supporting a separation of concerns making any application more efficient, modular and scalable.

- *Life is structured in layers. It is a structure that is both stable and flexible, consistent yet variable and it encompasses an infinite range of possibilities*

A Good Framework

- Designed to simplify development
  - Has already been built, tested, and industry hardened
  - Increases reliability and reduces programming time
  - Adheres to DRY principle
  - Helps enforce best practices and rules
  - Standards Based
  - Design Pattern based

Spring Framework
Based on Java Standards

JSR 330: Dependency Injection for Java

JSR-250 Common Annotations for the Java™ Platform

JSR-107 Annotations

JSR-303 - 349 Validation

JSR-352 Batch

JSR-107 JCache annotations

JSR-299 @Decorator and @Delegate

JSR-160 Connectors

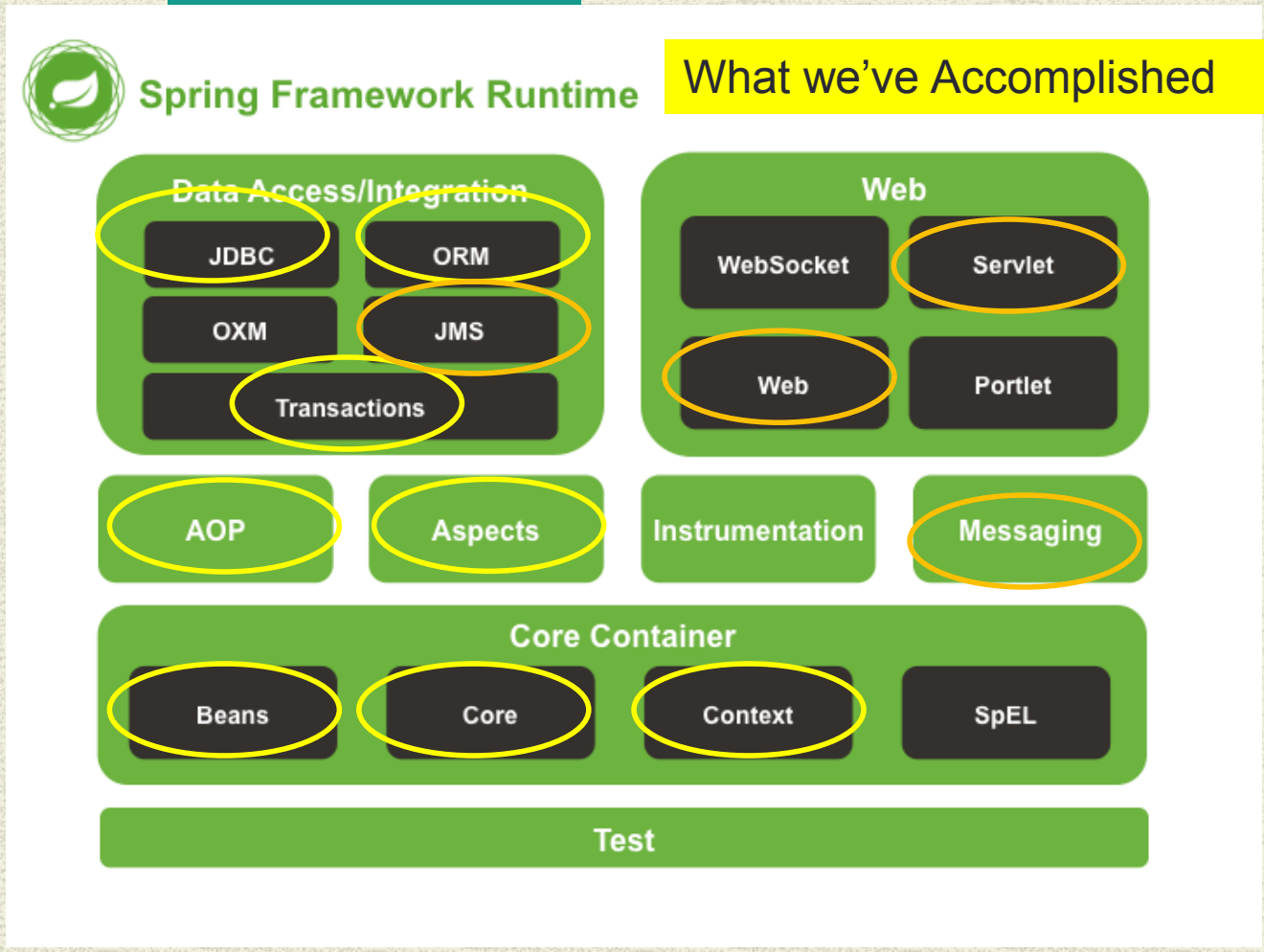
JSR 286 Portlets

Spring Framework Based on Design Patterns

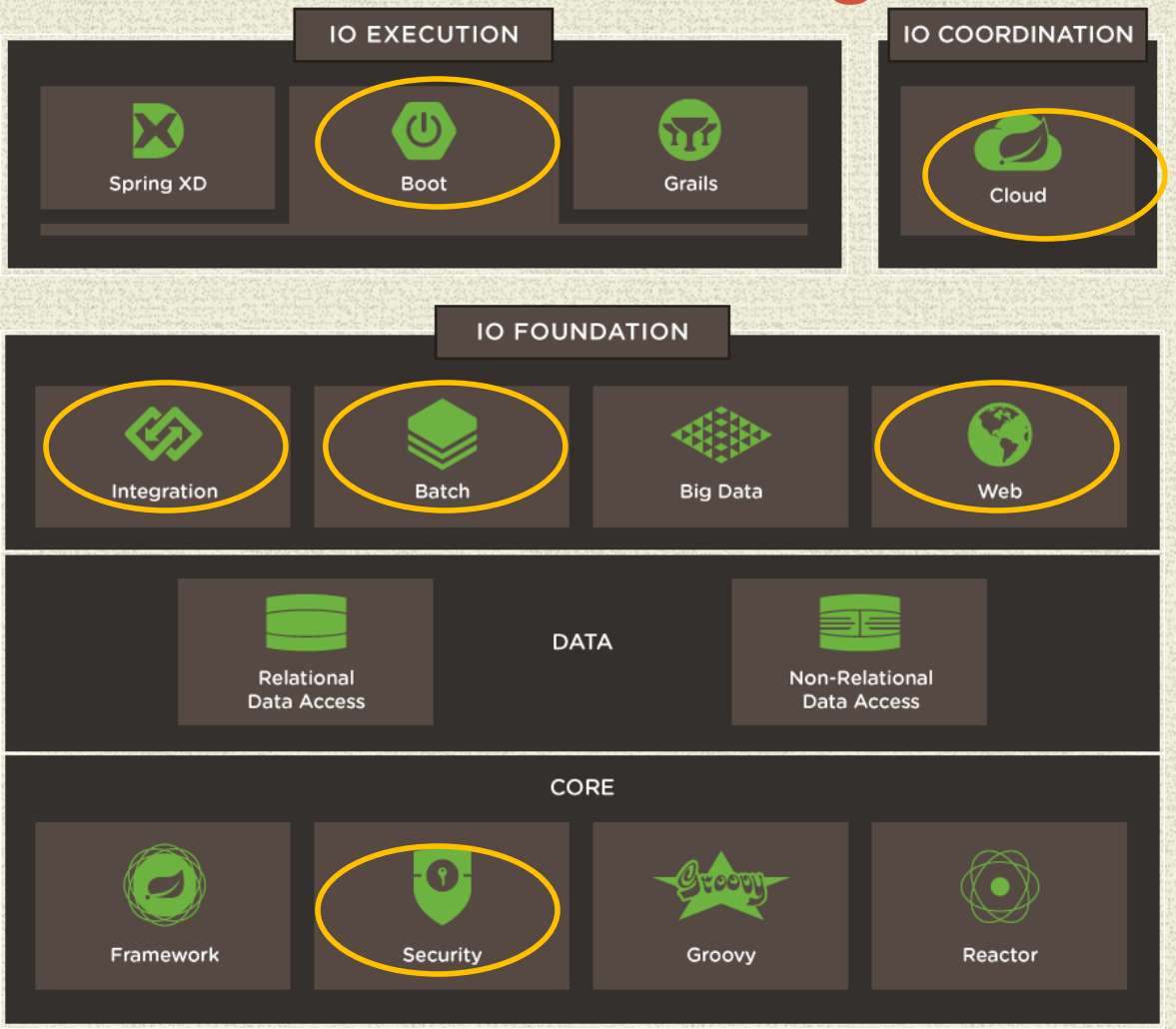
- Factory
- Proxy
- Singleton
- MVC
- Front Controller
- Template method
- Adapter
- Decorator
- Observer
- Interpreter
- Builder
- Factory method
- Abstract factory
- Composite
- Strategy
- Prototype
- Object pool
- Facade

Current Course [CS 544] Context

Coming Events



Additional Coming Events



Main Point

- Frameworks make development easier and more effective by providing a secure and reliable foundation on which to build upon.

- *The simplest form of awareness, Transcendental Consciousness, provides a strong foundation for a rewarding and successful life.*