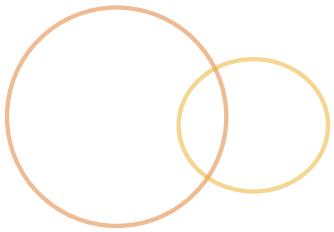
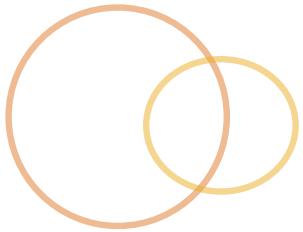
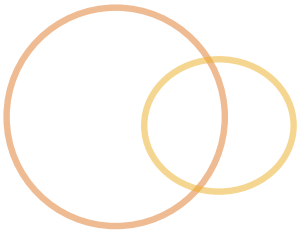
Introduction to Spring





Overview of Spring

Objectives



When we are done, you should be able to:

- Explain what Spring is
- Understand the issues of Java EE that Spring intended to fix

Java EE

- Allows distributed programming and is designed to be in large scale environments
- Overly complex
- Written for server, not ease of programming
- Difficult to unit test
- Tedious to configure
- Different on each tier

Note: This is especially true before Java EE 6

Impetus for Spring

- Writing Java EE should be easier
- It is better to program against interfaces than implementation
- Configuration should be easy
- OO design is more important than the underlying framework
- Testing is essential to quality software

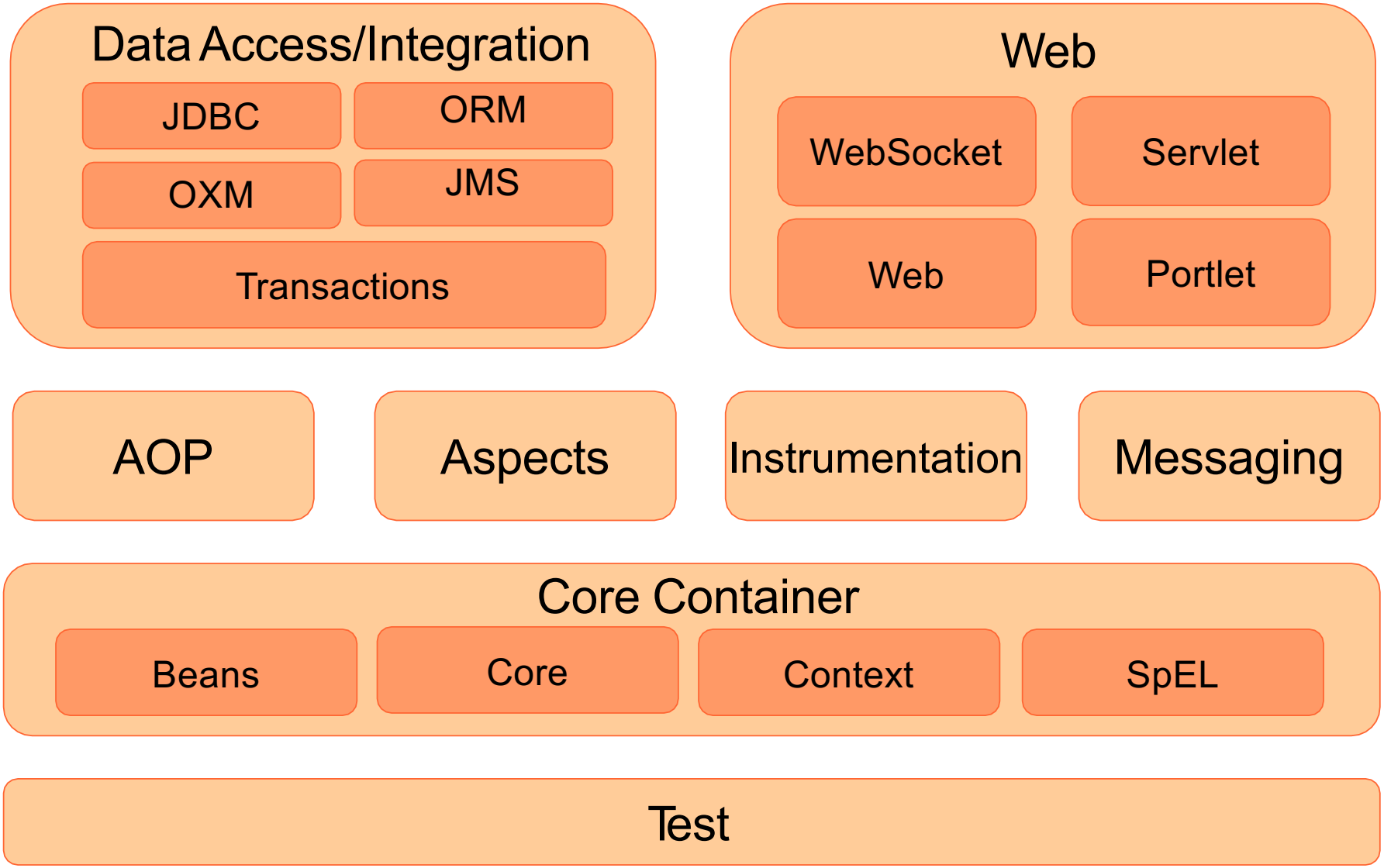
What is Spring?

- Spring is
  - A framework for consistent configuration of Java applications
  - A Dependency Injection Container
- The goals of Spring are to
  - Provide consistent infrastructural support
  - Encourage two basic design principles
    - DRY (Don't Repeat Yourself)
    - SoCs (Separation of Concerns)
  - Encourage tight cohesion and loose coupling

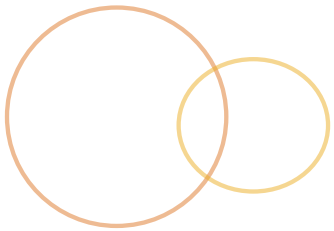
Requirements

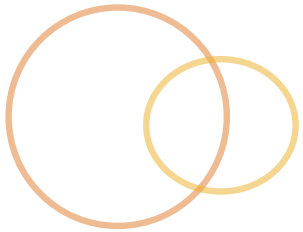
- General Requirements
  - Version of Java usable by version of Spring
    - Spring 4.x needs Java SE 1.7 or higher
  - Core Spring Framework Libraries
    - Consists of about 20 modules

Spring Framework Runtime



The Spring Container

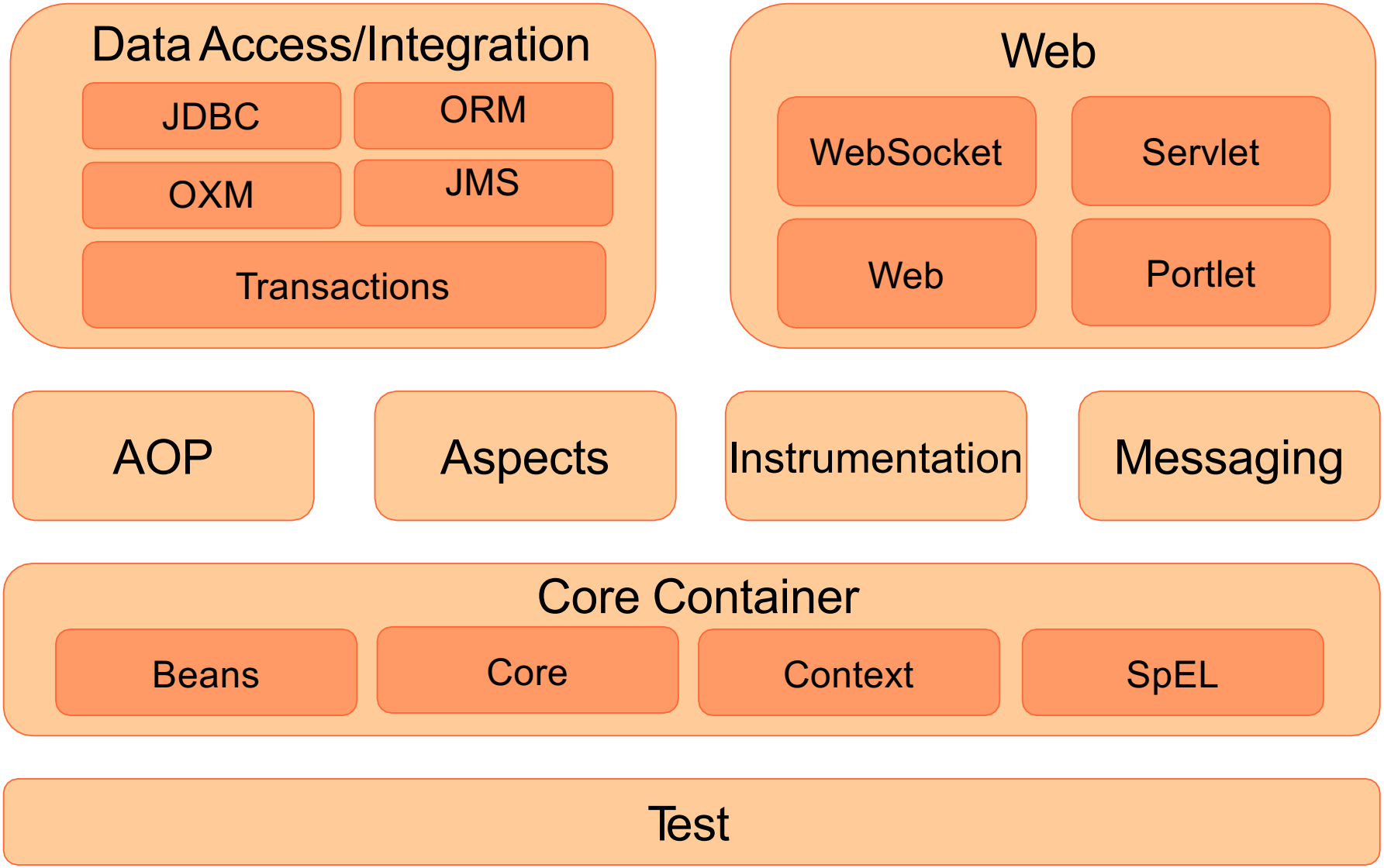




Spring Container

- Manages lifecycles of beans
- Inserts proxies when appropriate
- Inversion of Control (IoC) container is the basis
  - a.k.a. Dependency Injection
  - Objects define their dependencies, container injects those dependencies when bean is created
- `org.springframework.context.ApplicationContext`
  - Basis for Spring's IoC

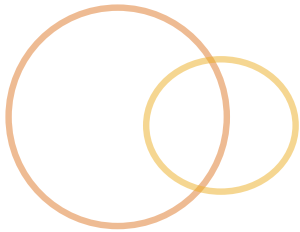
Spring Framework Runtime



Data Access/Integration

- `spring-jdbc`
  - JDBC abstraction Layer
- `spring-tx`
  - Support for programmatic and declarative transaction management
- `spring-orm`
  - Object-relational mapping for JPA, JDO and Hibernate
- `spring-oxm`
  - Supports Object/XML mapping
- `spring-jms`
  - Java Messaging Service features

Web Layer



- `spring-web`
  - Web integration features
  - Uses servlet listeners to initialize IoC
  - Contains HTTP client
- `spring-webmvc`
  - a.k.a. Web-Servlet module
  - Module-view-controller framework for web
  - REST web services

Web Layer [cont.]

- `spring-websocket`
  - Implementation of RFC 6455 and JSR-356
  - WebSocket protocol defines a two-way, client-server communication that is full duplex
- `spring-webmvc-portlet`
  - MVC implementation for a portlet environment
  - Mirrors functionality of `spring-webmvc`

Middle Layer

- `spring-aop`
  - Aspect-oriented programming (AOP) implementation
  - Based off of AspectJ
- `spring-aspects`
  - Actual integration with AspectJ
- `spring-instrument`
  - Automatic discovery of Managed Beans and applications
  - Provides runtime control of container
  - Exposes performance metrics and monitoring
- `spring-messaging`
  - Annotations for mapping messages to methods

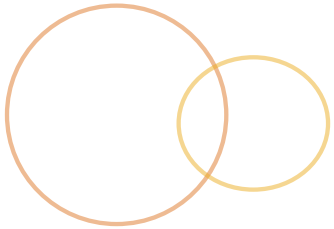
Core Container

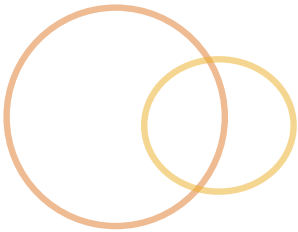
- `spring-core` and `spring-beans`
  - The basis of the framework
  - Provides IoC and dependency injection features
  - Allows decoupling of configuration and specification dependencies from the business logic
- `spring-context` and `spring-context-support`
  - Provides access to objects
  - `ApplicationContext` is from here
  - Supports internationalization, event propagation, resource loading and creation of contexts
  - Supports EJB, JMX and basic remoting

Core Container [cont.]

- `spring-expression`
  - Expression language based on unified expression language (EL)
  - Uses JSP 2.1 specification

Testing





- `spring-test`
  - Unit testing and integration testing with Junit or TestNG
  - Provides loading and caching of contexts
  - Provides mock objects

Lab 0 - Setup