Introduction to Spring Framework

CodeMash 2015

Pre-Compiler
Java: Beginner to Intermediate

David Lucas
Lucas Software Engineering, Inc.
www.lse.com
ddlucas@lse.com

Spring Framework: Introduction Agenda

- Who is the speaker ?
- What setup do we need ?
- Why dependency injection ?
- What are the basics of the Spring Framework?
    - Core
    - Data
    - Web
- What else?

Spring Framework: Introduction
Who is the speaker?

- David Lucas
- Lucas Software Engineering, Inc. (LSE) www.lse.com
- Focus on highly scalable Java Services
- Worked in various industries:
  military, insurance, financial, manufacturing, utilities
- Provide software engineering, performance tuning, mentoring and training
- Love to use SQuirreL Client

LSE

Spring Framework: Introduction
Ground Rules

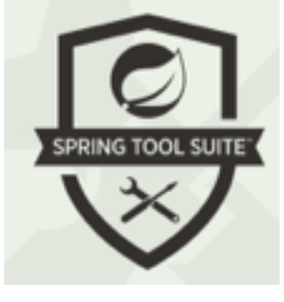
- You will not hurt my feelings if you …
  - ask questions
  - take your phone or text conversation outside
  - leave because you are bored
  - want me to change speed (slower / faster)
- This requires audience participation

Spring Framework: Introduction Assumptions

- This is an introduction, should not drill too deep into details
- Will not go into scripting or domain specific languages
- Will not dig into Spring Batch or Spring XD
- Will touch on basics of Spring Framework
- Will build examples using maven
- Examples are for information purposes
- Unit tests are not exhaustive

Spring Framework: Introduction
What setup do we need?

- Java JDK 1.8
- Spring Tool Suite 3.6.3
- Copy of examples
- Unzip examples and import projects into STS
- https://github.com/lseinc/intro-spring
- Verify Setup



Spring Framework: Introduction Dependencies

- Are dependencies good ?
- Why we like them ?
- Why we don’t like them?
- What patterns do we end up using?
  - singleton
  - factory
  - service locator
  - push versus pull

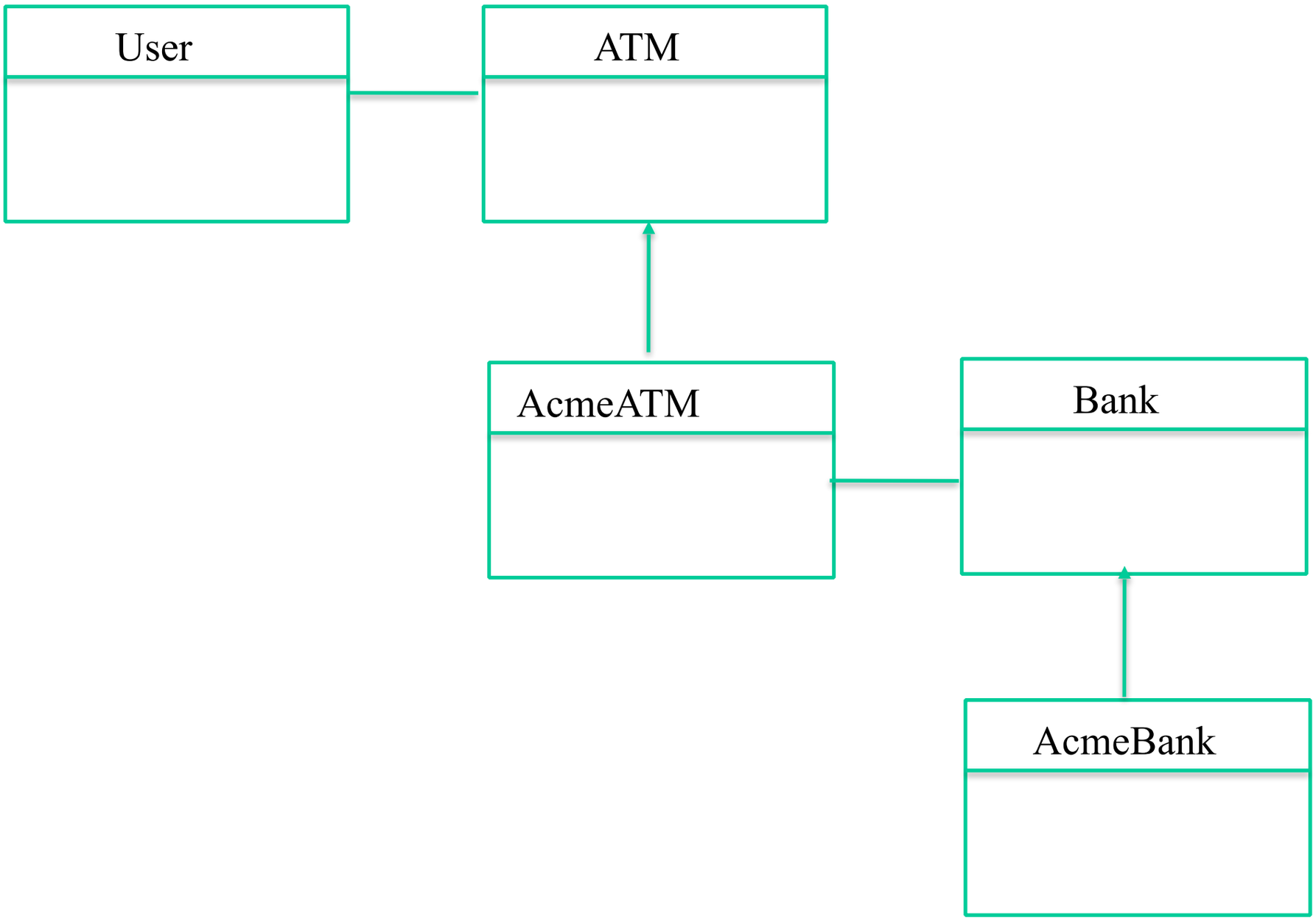
Spring Framework: Introduction Core

- What does POJO dependencies look like?
  Lab 01
- What does it look like with Spring XML ?
  Lab 02
- What does it look like with Spring Annotations?
  Lab 03
- What are Spring profiles?
  Lab 04 / Lab 05

Spring Framework: Introduction Dependencies

- LAB 01 simple POJO dependencies
  - Go to unit test and uncomment the Asserts
  - Change code to make test green
  - Create the HAS-A relationships between User to ATM to Bank

LAB 01: Model



LAB 01

Wait here until ready to move on…

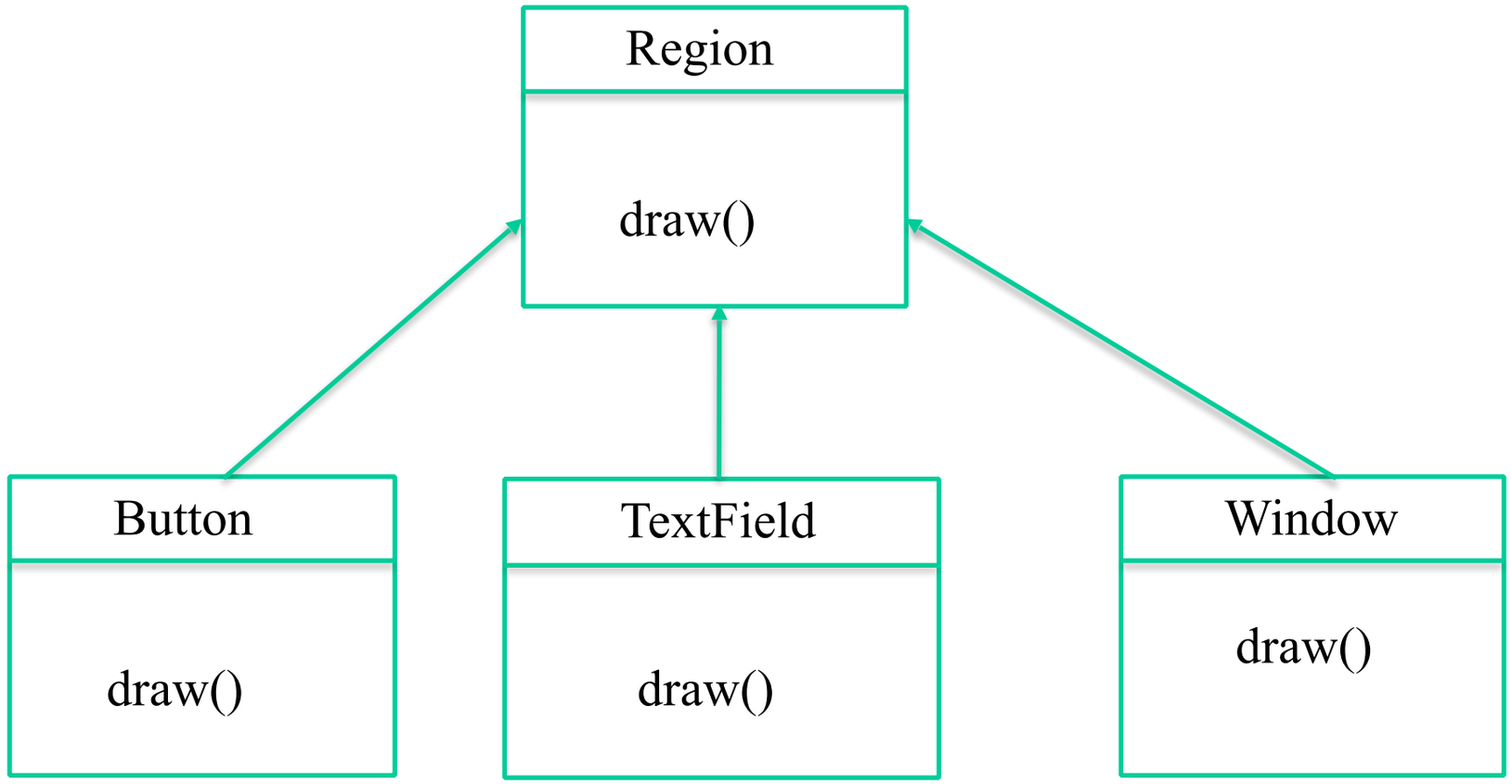
Spring Framework: Introduction

- What are frameworks?
- What are containers?
- What is IOC?
- What is the Spring Framework?
- Core Components
- Some examples

What are Frameworks?

- Software library that provides reusable API
- An abstraction providing generic functionality that can be extended to provide specialized features
- owns flow of control and/or lifecycle
- default behavior that is usable
- examples: MFC, X11, Qt, Eclipse RCP, ASP.NET, EJB, JSP / Servlets, Hibernate, etc… and of course, Spring

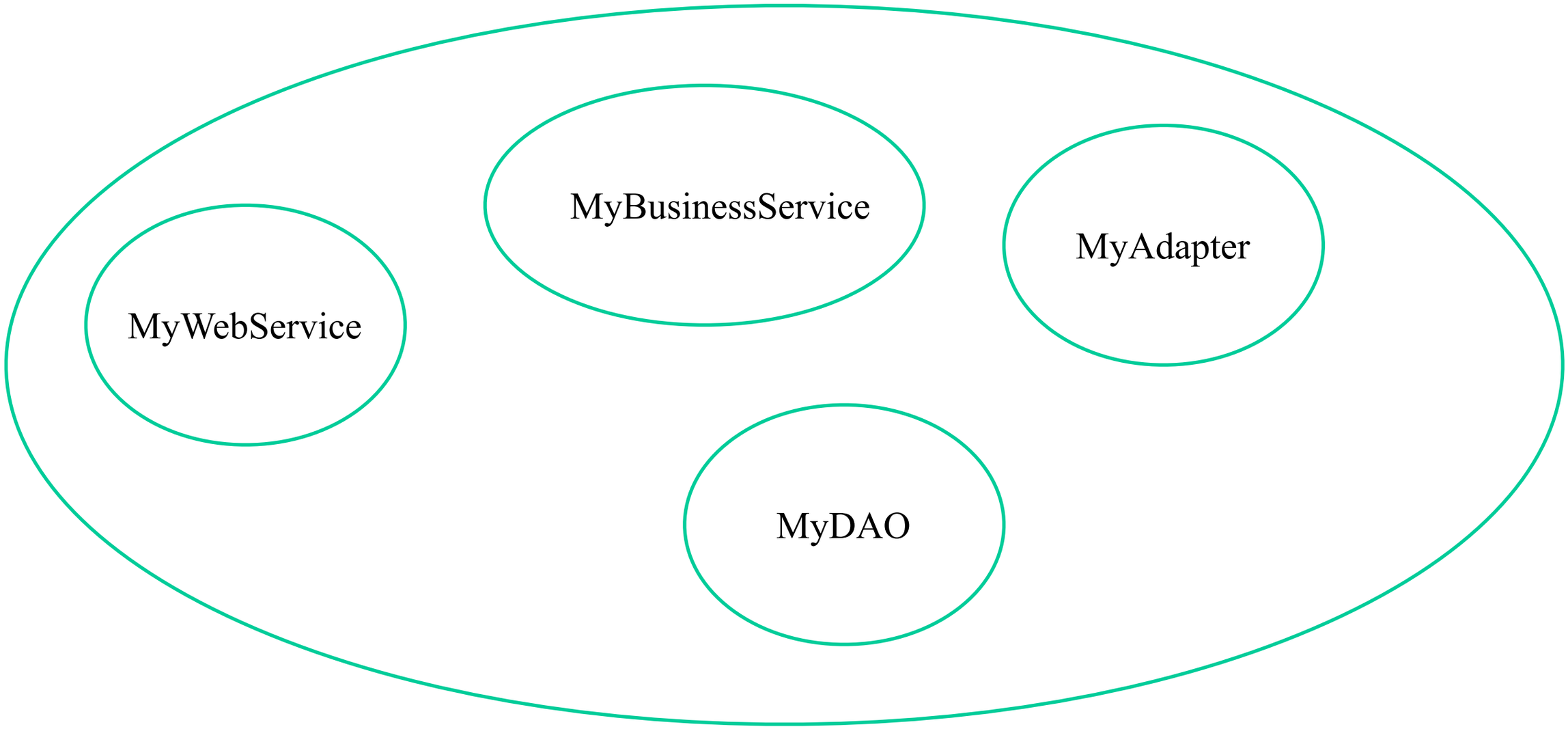
Example Framework



What are Containers?

- an environment that contains and manages the access to one or more objects
- collections are simple containers
- more complex containers manage the lifecycle of objects
- containers can also handle configuration dependencies
- dependency injection takes creation away from the constructor
- examples: EJB Container, Web Container

Example Container



What is IOC?

- IOC is inversion of control (implicit versus explicit)
- Martin Fowler compared it to the
  Hollywood Principle, “Don’t call us, we’ll call you !”
- You don’t call the APIs directly,
  the container framework calls you
- Leverage existing template patterns
- A subset is known as Dependency Injection (DI),
  lifecycle of object part of framework
- dependent object configuration and control are
  taken away from the typical construction
- containers manage wiring of dependencies

Why IOC?

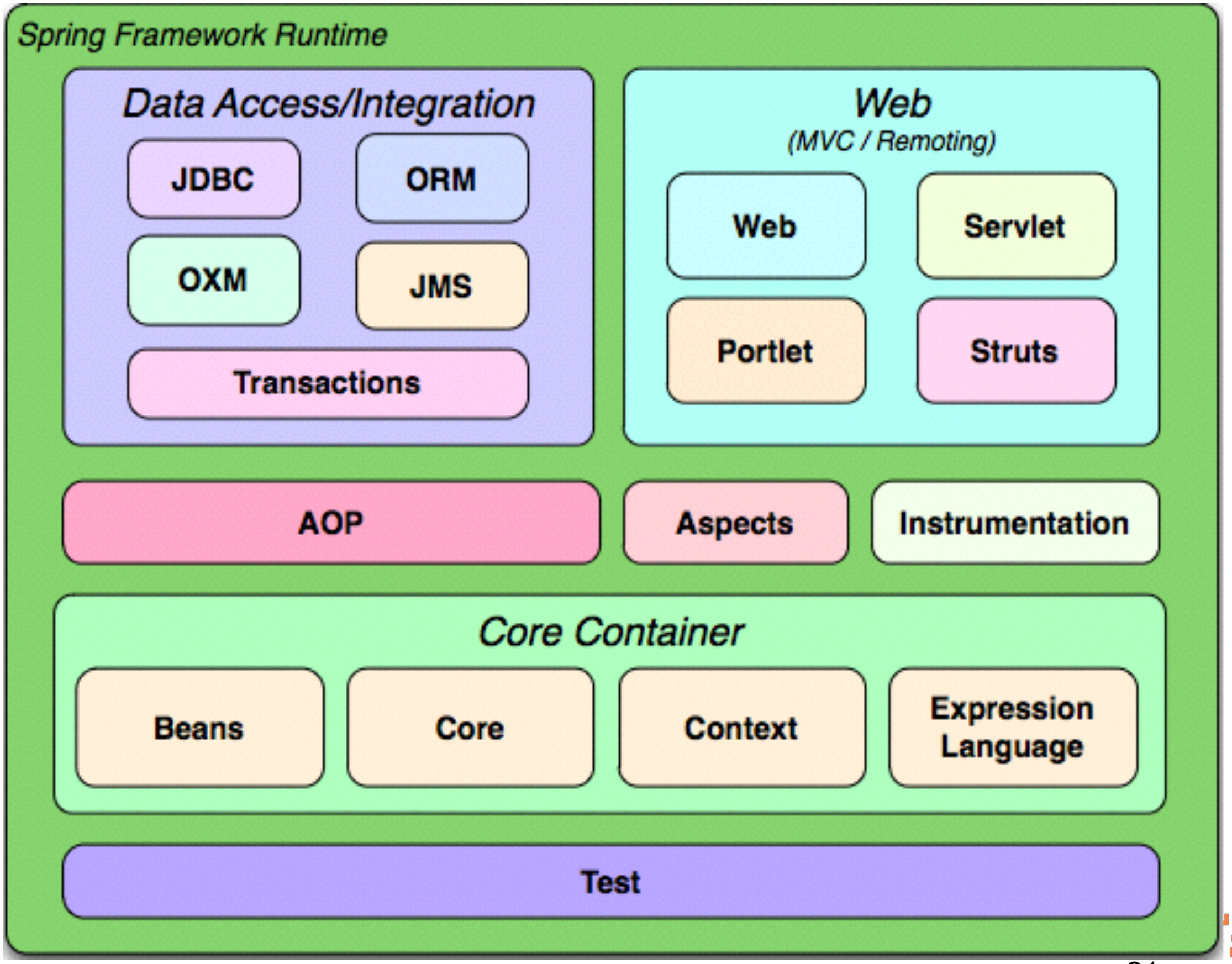
- Inversion of Control removes direct dependencies (no concrete knowledge of implementation)
- The most flexible systems make change additive.
- Avoid modifying code.
- Configuration can be abstracted.
- Life Cycle Management:
  patterns: singleton, prototype, factory, service locator
- Loose coupling
- EASIER TO TEST !

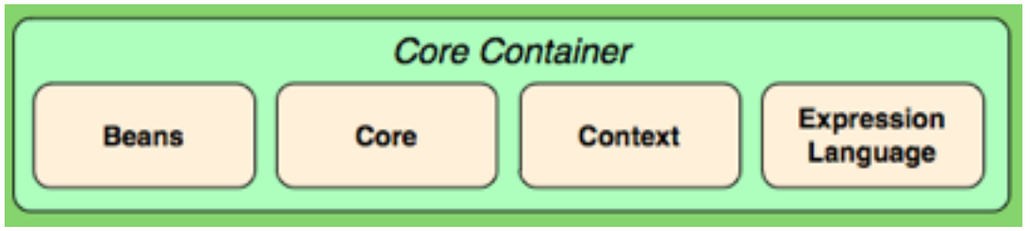
What is the Spring Framework?

- Created in 2002-2003 by Rod Johnson
- goal: to simplify java development
- Invented out of necessity; J2EE was still overly complex and hard to assemble
- pico container that manages beans (pojo: plain old java object)
- leverages common patterns in a framework to reduce developers re-inventing the wheel
- Apache License 2.0 (open source)

What is the Spring Framework?

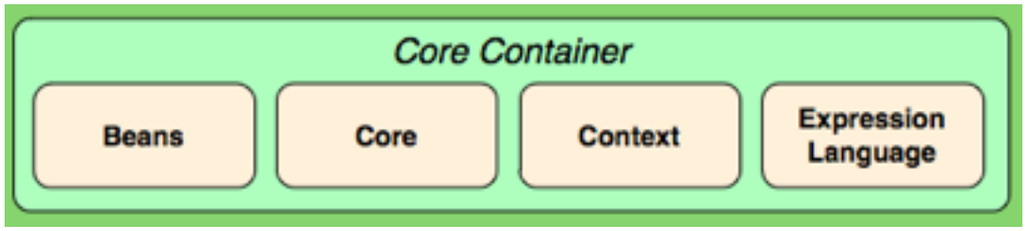
- Spring is not focused on implementations, but wiring implementations together
- provides “context” to a set of configured beans and properties
- provides dependency injection (DI) to decouple implementations
- provides cross-cutting of concerns (aspects and interceptors)
- leverages proxies to extend objects at runtime and delegate to targets


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



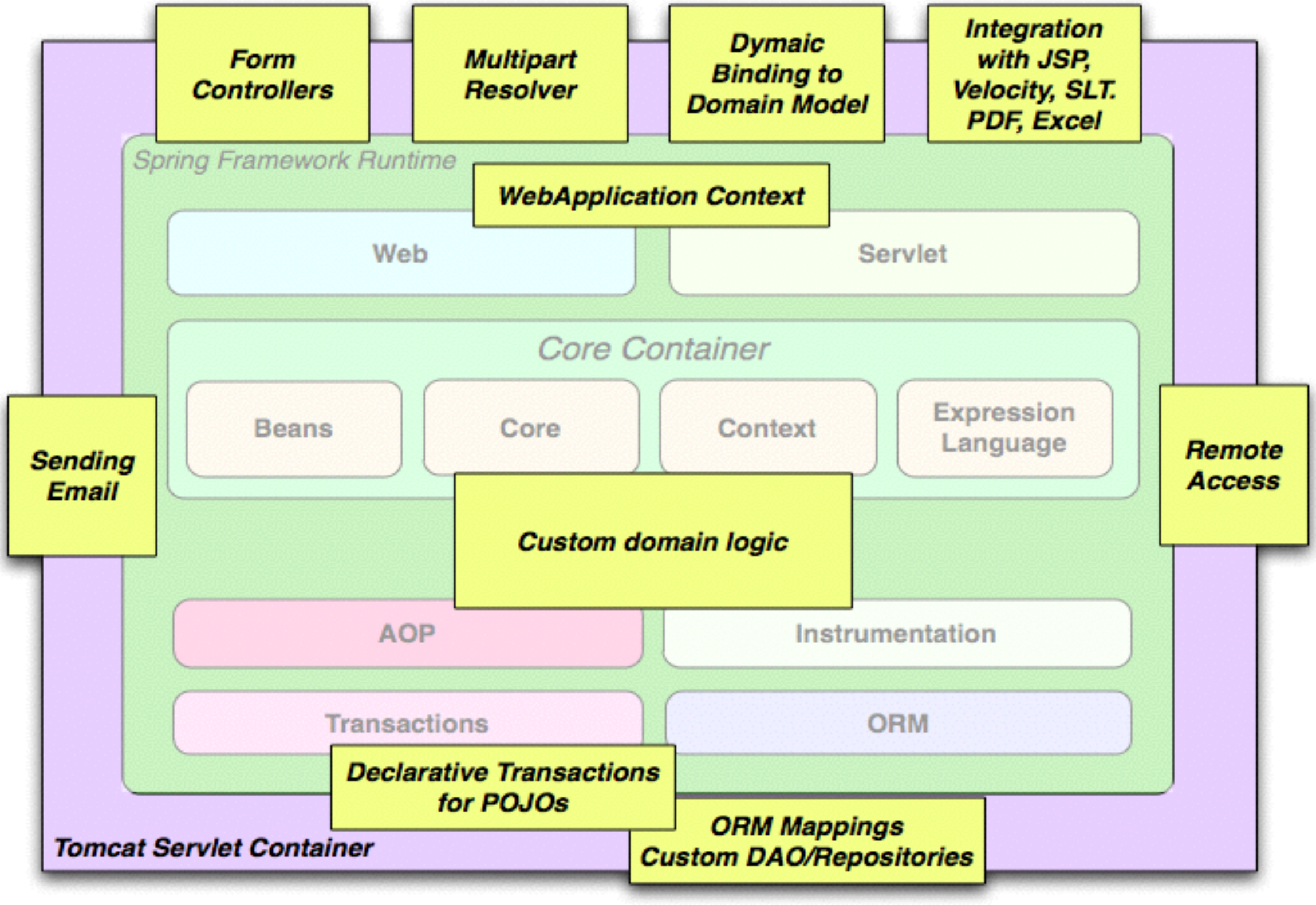
Spring Core

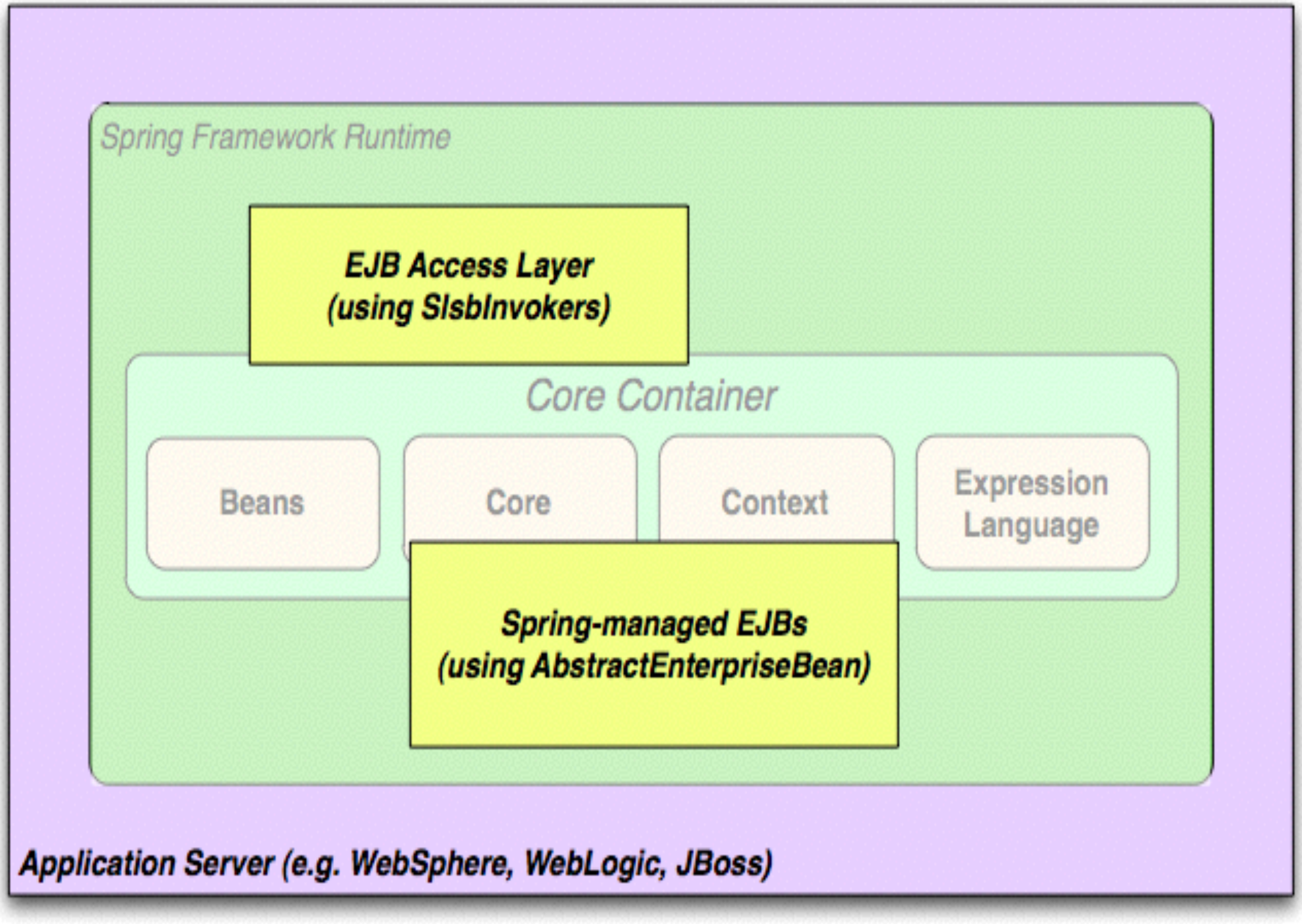
- http://docs.spring.io/spring/docs/current/spring-framework-reference/html/beans.html
- Core Container are the common objects leveraged in all the extensions
- Core consists of Core, Beans, Context, and Expression Language modules
- Core and Beans modules are focused on the IOC and Dependency Injection features as well as life cycle management using the BeanFactory
- Context module provides the container the ability

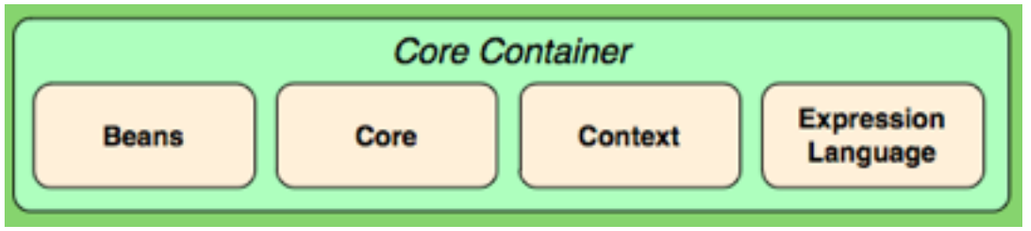


Spring Core

- Context module adds
  - internationalization
  - property configuration.
- Context integrates with Java EE containers like:
  - EJB
  - JMX
  - Web Containers

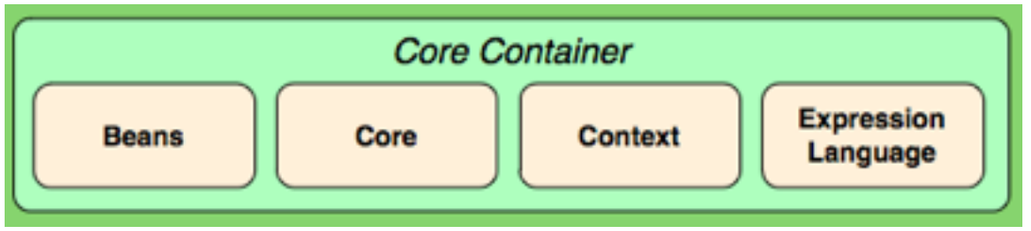

Form Controllers
Multipart Resolver
Dymaic Binding to Domain Model
Integration with JSP, Velocity, SLT. PDF, Excel
Spring Framework Runtime
WebApplication Context
Web
Servlet
Core Container
Beans
Core
Context
Expression Language
Sending Email
Remote Access
Custom domain logic
AOP
Instrumentation
Transactions
ORM
Declarative Transactions for POJOs
ORM Mappings Custom DAO/Repositories
Tomcat Servlet Container


Spring Framework Runtime
EJB Access Layer
(using SlsbInvokers)
Core Container
Beans
Core
Context
Expression
Language
Spring-managed EJBs
(using AbstractEnterpriseBean)
Application Server (e.g. WebSphere, WebLogic, JBoss)



Spring Core

- Context is about the caller's point of view
- Beans are created based on a context's lifecycle
- Spring IOC uses two types of Dependency Injection (DI)
  - Type 2 (property)
  - Type 3 (constructor) creation of dependencies



Spring Core

- Bean Lifecycle
  - scope (instances)
  - initialization (callbacks)
  - finalization (callbacks)

Spring Framework: LifeCycle

- IOC Type 2 setters
  - Java Bean Property (requires getter / setter)
  - Injected via
    - XML: via property name/value of bean
    - @Autowired
- IOC Type 3 constructor
  - Java Bean public constructor
  - Injected via
    - XML via constructor-arg elements
    - @Autowired on parameters

Spring Framework: Introduction Beans

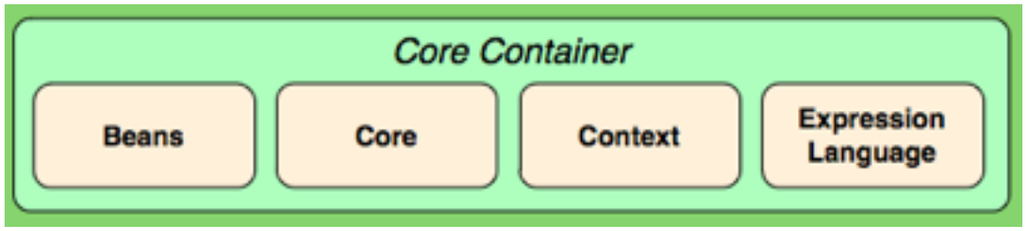
- Beans have scope (singleton , prototype)
  Just like Web has request, session, application

```
<bean id="accountDao"
      lazy-init="true" scope="singleton"
      class="AccountDaoJDBC">
      <!-- ... -->
</bean>
```

Spring Framework: LifeCycle

- Beans have lifecycle
  - InitializingBean and DisposableBean
  - @PostConstruct / @PreDestroy
  - Lifecycle (interfaces for events)

```
public class AccountDaoJDBC implements AccountDao,
                    InitializingBean, DisposableBean{
    public void afterPropertiesSet() throws Exception {}
    public void destroy() throws Exception {}
    //...
}
```



Spring Core

- Bean Properties
  - conversion (String to double)
  - validation (check for nulls)
  - formatting (String formats)

Spring Framework: Introduction Beans

- Example Service Implementation

```
public class AcmeBank {
   //…
  private String name;
   //…
}
```

Spring Framework: Introduction Beans

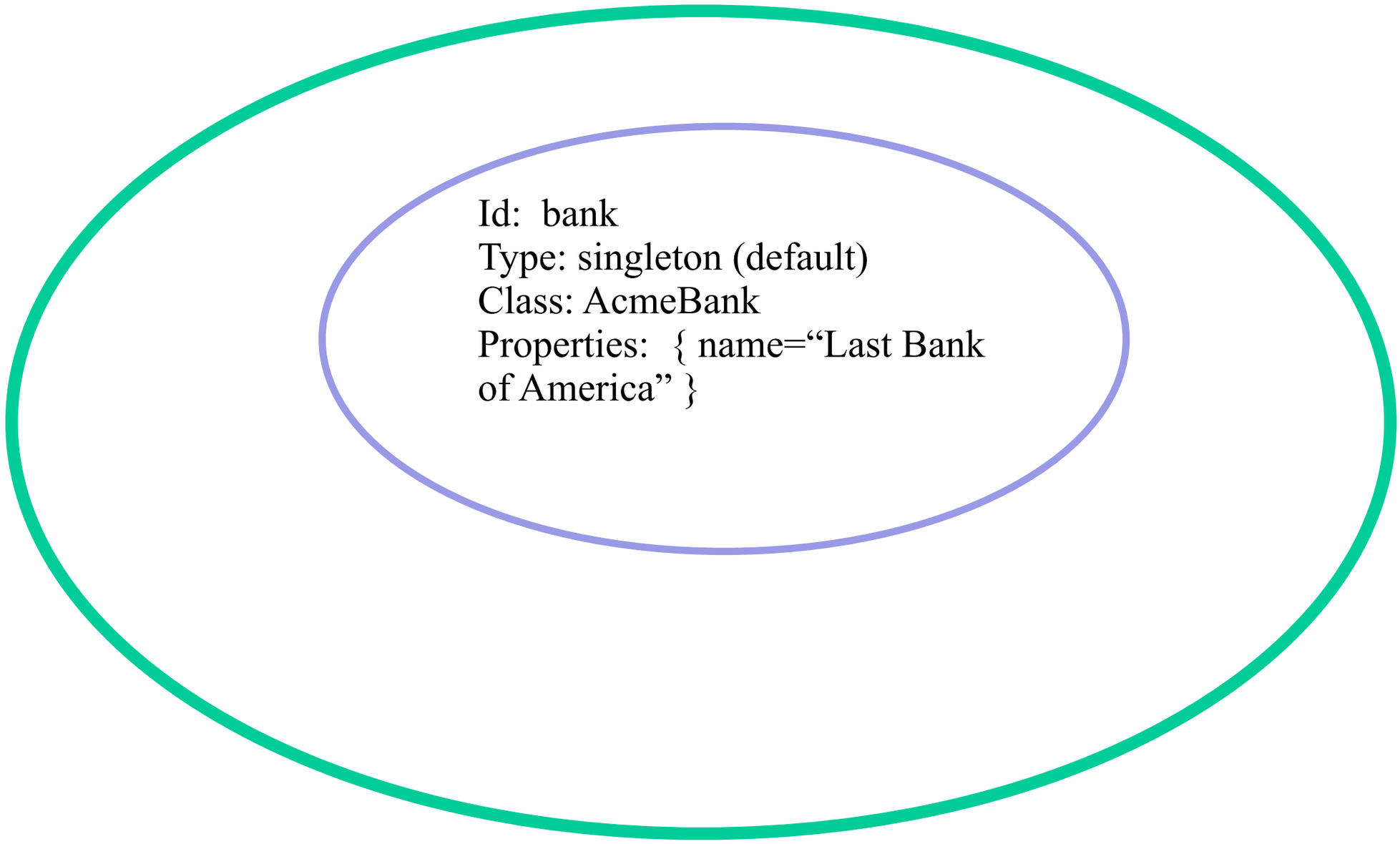
- XML Application Context
  - Uses XSD to define capabilities
  - http://docs.spring.io/spring/docs/current/spring-framework-reference/html/xsd-config.html

```
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/context
      http://www.springframework.org/schema/context/spring-context.xsd
      http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="bank" class="AcmeBankImpl">
        <property name="name" value="Last Bank of America=" />
    </bean>

</beans>
```

Application Container



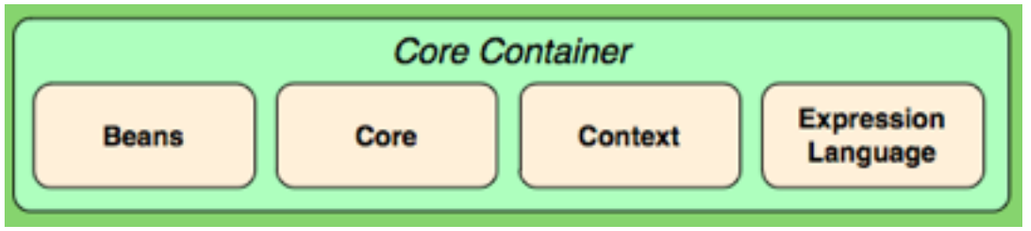
Spring Framework Beans

- Factory Methods
  - static method to create object instance

```
<bean
    id="clientService"
    class="examples.ClientService"
    factory-method="createInstance"/>
```

  - instance reference and factory method name

```
<bean
    id="serviceLocator"
    class="examples.DefaultServiceLocator"/>
<bean
    id="clientService"
    factory-bean="serviceLocator"
    factory-method="createClientServiceInstance"/>
```



Spring Core

- Example of Expression Language Usage:

```
<bean class="mycompany.RewardsTestDatabase">
    <property name="databaseName"
        value="#{systemProperties.databaseName}" />
    <property name="keyGenerator"
        value="#{strategyBean.databaseKeyGenerator}" />
    <!— .... —>
</bean>
```

Spring Framework: Introduction Dependencies

- Spring Beans Maven Configuration
  - properties
    - via PropertyConfiguration
    - via Environment
  - dependencies
    - via explicit depends on attribute
    - via implicit references to other beans

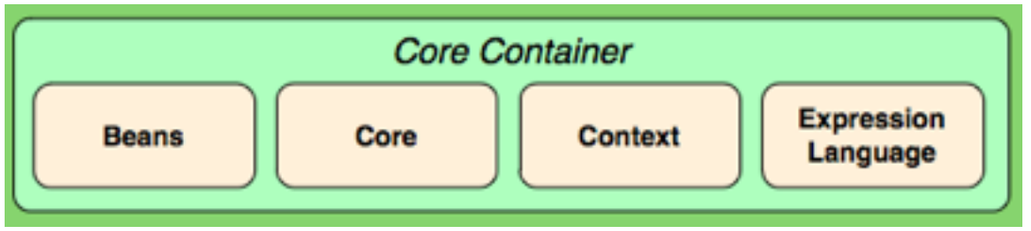
Spring Framework: Introduction Dependencies

- Enable Spring Configured Annotations (Using @Configuration on a class, the class gets injections on @Autowired and @Resource)

```
<context:spring-configured/>
```

- Enable Spring Properties (${prop.key} key references in XML and @Value on fields are injected with property value)

```
<context:property-placeholder
    location="classpath:application.properties"/>
```



Spring Core

- Expression Language module provides a unified EL as specified in JSP 2.1
- EL supports setting/getting property values, property assignments, method invocations, and conversions of types (lists, maps)
- Spring supports configuration via XML Configuration file or via @Annotations

Spring Framework: Introduction Core Annotations

- DZone Reference Card (very handy)
  - Spring Configuration
    http://refcardz.dzone.com/refcardz/spring-configuration
  - Spring Annotations
    http://refcardz.dzone.com/refcardz/spring-annotations

Spring Framework: Introduction Dependencies

- LAB 02 wire POJO with Spring XML
  - Go to unit test and uncomment the Asserts
  - Change XML to make test green
  - Create the HAS-A relationships between User to ATM to Bank

LAB 02

Wait here until ready to move on…

Spring Framework: Introduction Dependencies

- Enable Spring Configured Annotations (Using @Configuration on a class, the class gets injections on @Autowired and @Resource)

```
<context:spring-configured/>
```

- Enable Spring Properties (${prop.key} key references in XML and @Value on fields are injected with property value)

```
<context:property-placeholder
    location="classpath:application.properties"/>
```

Spring Framework: Introduction
Core Annotations

- Injections
  @Autowired
  @Value
- Component Beans
  @Service
  @Repository
  @Component

Spring Framework: Introduction
Core Annotations

- Bean Configurations
  @Configuration
  @ComponentScan
  @PropertySource
  @EnableAutoConfiguration
  @Import
  @EnableTransactionManagement

Spring Framework: Introduction Dependencies

- LAB 03 wire POJO with Spring Annotations
  - Go to unit test and uncomment the Asserts
  - Change Objects to make test green
  - Create the HAS-A relationships between User to ATM to Bank

LAB 03

Wait here until ready to move on…

Spring Framework: Introduction Core Annotations

- On Component Classes
  @Profile("acme-bank")
- On JUnit Test Class
  @ActiveProfiles("acme-bank")
- java -Dspring.profiles.active=acme-bank argument to set from command line application
- XML Config uses a beans wrapper with profile attribute

```
<beans profile="acme-bank" >
  <bean id="theBank"
      class="com.lse.spring.example.pojo.AcmeBank"
      scope="singleton" >
      <property name="checkingBalance"
                value="${my.bank.balance}" />
      <property name="bankName"
                value="${my.bank.name}" />
  </bean>
</beans>
```

Spring Framework: Profiles

- LAB 04 use profiles with Spring XML
  - Change old Bank Test to use a profile
  - Add new Bank Test with new profile
  - Add new Bank and wire it based on profile setting
- Change Objects to make test green

LAB 04

Wait here until ready to move on…

Spring Framework: Profiles

- LAB 05 use profiles with Spring Annotations
  - Change old Bank Test to use a profile
  - Add new Bank Test with new profile
  - Add new Bank and wire it based on profile setting
- Change Objects to make test green

LAB 05

Wait here until ready to move on…

Spring Framework: Introduction Data

- Need a database: Derby on the back stretch !
  Lab 06
- Spring JDBC
  Lab 07
- Spring Transactions
  Lab 08
- Spring JPA
  Lab 09
- Spring Boot

Spring Framework:
LAB 06 - Setup Derby

- perform maven clean install on derby project
- perform maven run on derby project
- should see java process listening on port 1527
- setup STS to connect to database
- either run the sql scripts under src/main/resources
  or unzip provided derby-data zip file

LAB 06

Wait here until ready to move on…

Spring Framework: Data

- DAO: data access object (data layer)
- Spring provides consistent access to JDBC interfaces to include JDO, JPA, Hibernate, iBatis templates.
- Spring provides common exception hierarchy

Spring Framework: Data

- @Repository configures DAO with common exception translation
- JDBC Template (Spring JDBC)
- iBatis SqlMapClient support
- Hibernate Session API support
- JPA supported entity manager (EJB)
  - @PersistenceContext
  - @PersistenceUnit

Spring Framework: Data

- JdbcTemplate uses DataSource provided by Spring Application Context
- DAO uses JdbcTemplate to execute JDBC type commands and manages closing of connections, statements, and result sets
- update method is for executing SQL that performs insert, update, or delete operations
- query executes selects
- Results use a DataMapper to do manual mapping of each row to new entity object

Spring Framework: Data

- DataSource leveraged via XML configuration

JNDI setup:

```
<jee:jndi-lookup id="dataSource" jndi-name="java:comp/env/jdbc/myds"/>

 <bean id="accountDao" lazy-init="true" scope="singleton"
      class="com.lse.spring.example.dao.AccountDaoJdbc">
            <property name="dataSource" ref="dataSource" />
</bean>
```

Spring Framework: Data

- DataSource leveraged via XML configuration DBCP setup:

```
<bean id="dataSource"
    class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
    <property name="driverClassName" value="${jdbc.driverClassName}"/>
    <property name="url" value="${jdbc.url}"/>
    <property name="username" value="${jdbc.username}"/>
    <property name="password" value="${jdbc.password}"/>
</bean>
<bean id="accountDao" lazy-init="true" scope="singleton"
            class="com.lse.spring.example.dao.AccountDaoJdbc">
    <property name="dataSource" ref="dataSource" />
</bean>
```

Spring Framework: Data

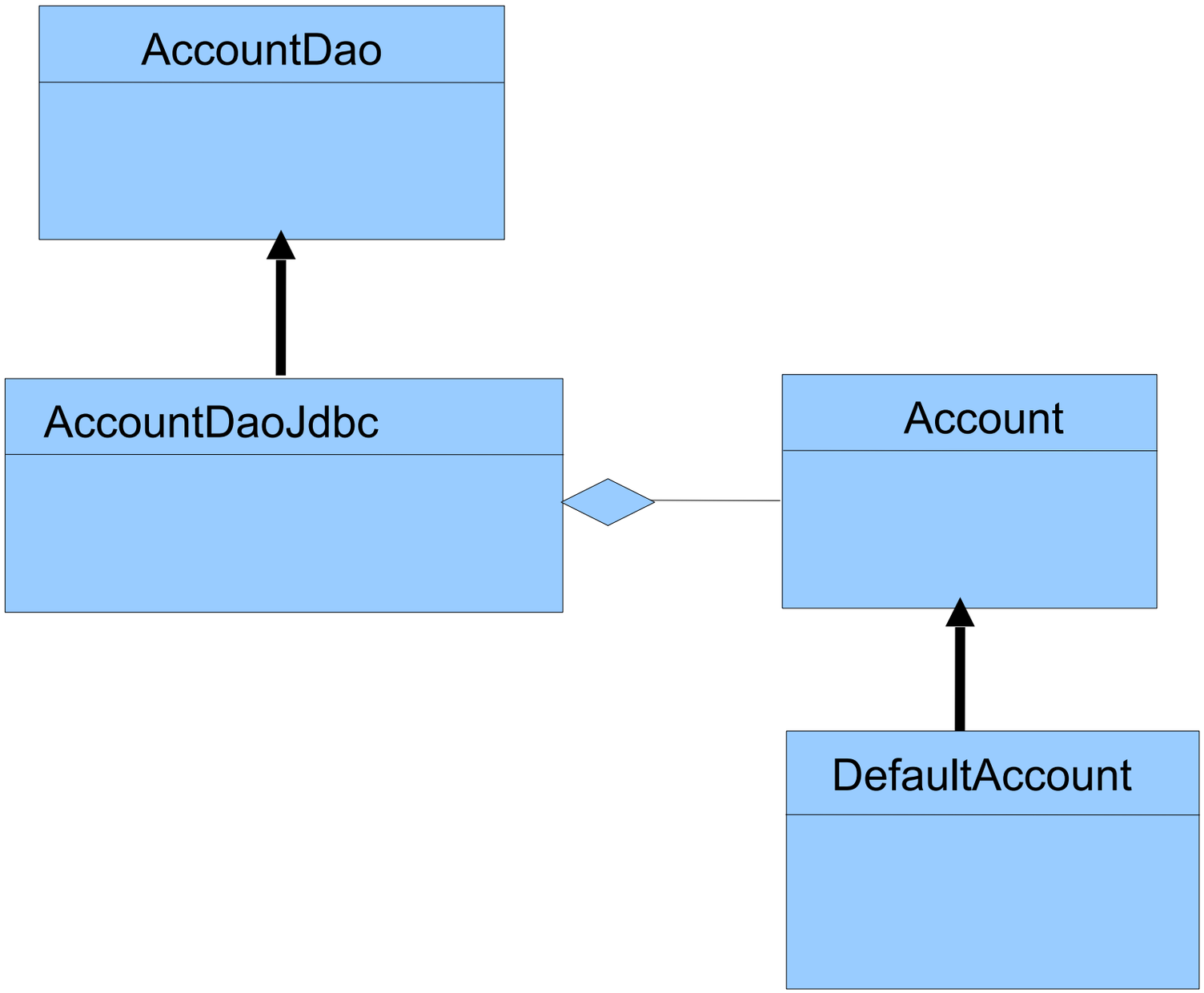
- Persistence
  - JDBC
  - Transactions
    - Local (JDBC)
    - Global (JTA)
- ORM
    - Hibernate
    - MyBatis
    - JPA (example: EclipseLink / Hibernate)
    - NoSQL

Spring Framework: Data - JDBC

- LAB 07 use Spring JDBC Template to create an AccountDAO implementation
  - Uncomment Asserts in AccountDAOTest
  - Uncomment Asserts in UserTest
  - Change Objects to make test green

LAB 07: Data JDBC

Hint to model::

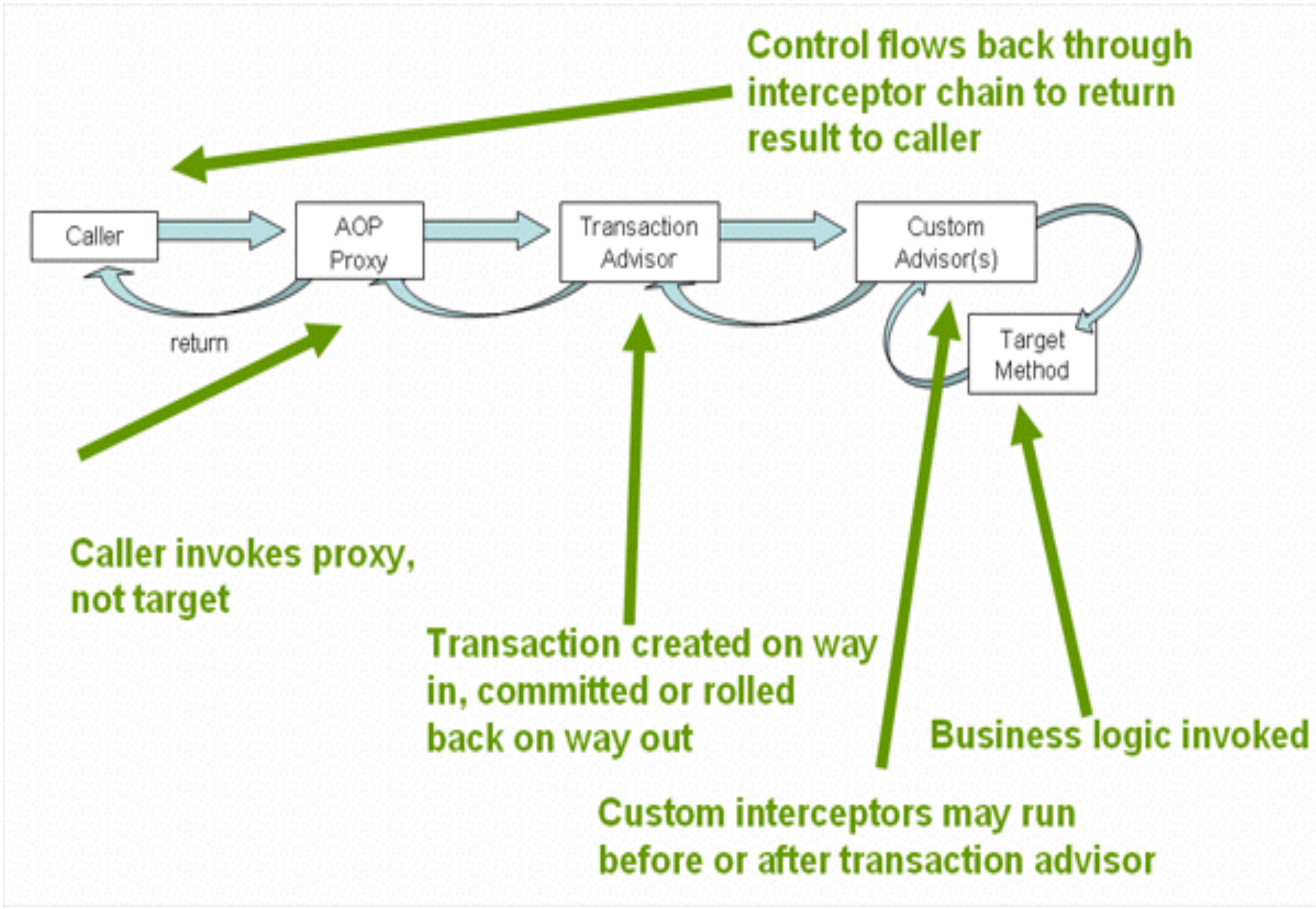


LAB 07

Wait here until ready to move on…

Spring Framework: Data Transactions (TX)

- Spring Framework provides Transaction Management integration
- Leverages application container TMs: WebLogic, WebSphere, JBoss, etc
- Provides Local and Global support
  - JTA: Java Transaction API (XA support)
- Spring provides consistent transaction management for multiple resources
  - database (JDBC, JPA)
  - messaging (JMS)
  - RMI / IIOP


Control flows back through interceptor chain to return result to caller
Caller
AOP Proxy
Transaction Advisor
Custom Advisor(s)
Target Method
return
Caller invokes proxy, not target
Transaction created on way in, committed or rolled back on way out
Custom interceptors may run before or after transaction advisor
Business logic invoked

Spring Framework: Data Transactions

- Simple DataSource TransactionManager
- Not Two Phase Commit
- Only Supports JDBC transactions
- Local Transaction Support (not Global)

```
<!-- a PlatformTransactionManager is still required -->

<bean id="transactionManager"
  class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <!-- (this dependency is defined somewhere else) -->
  <property name="dataSource" ref="dataSource"/>

</bean>
```

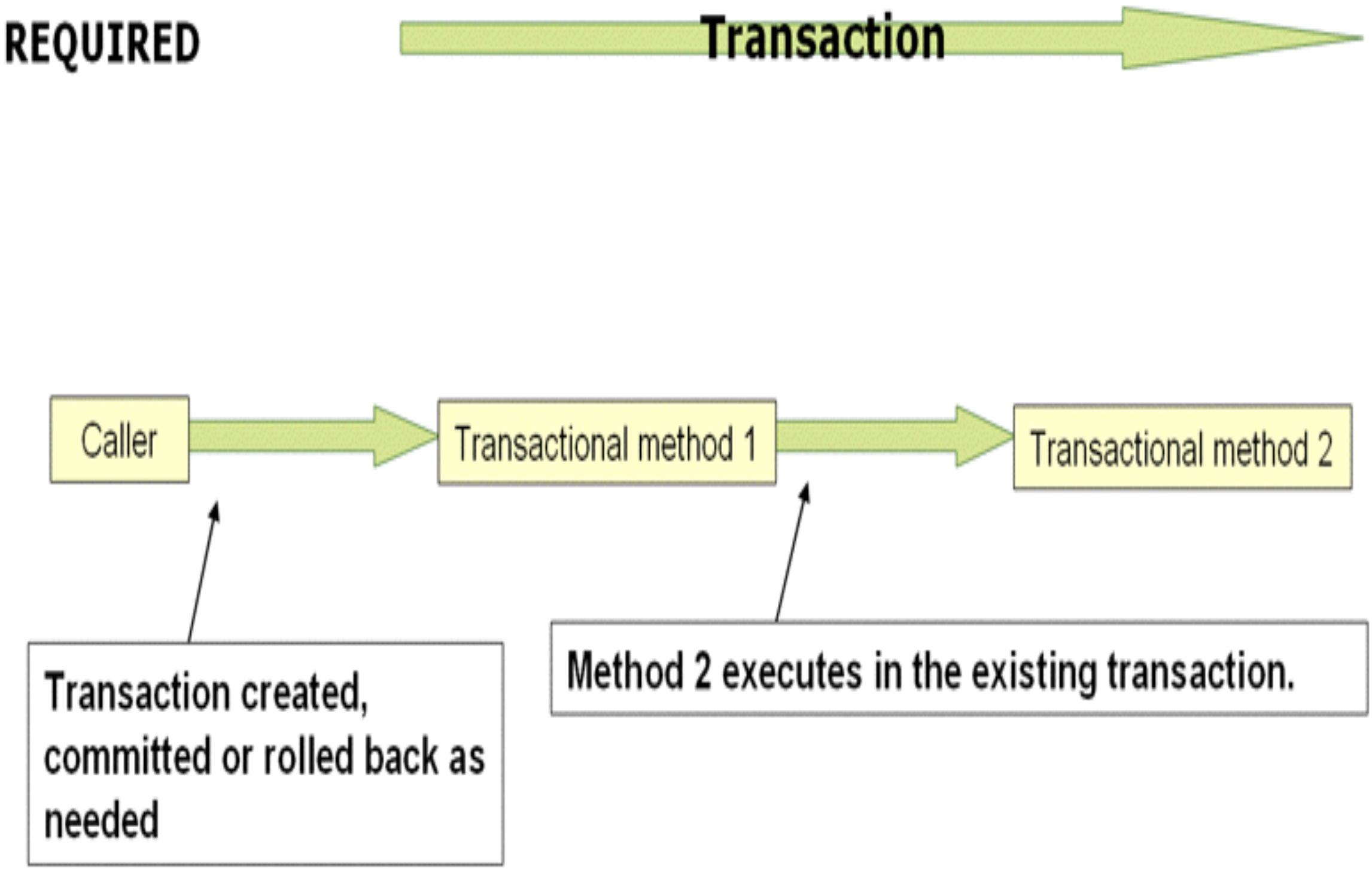
Spring Framework: Data Transactions

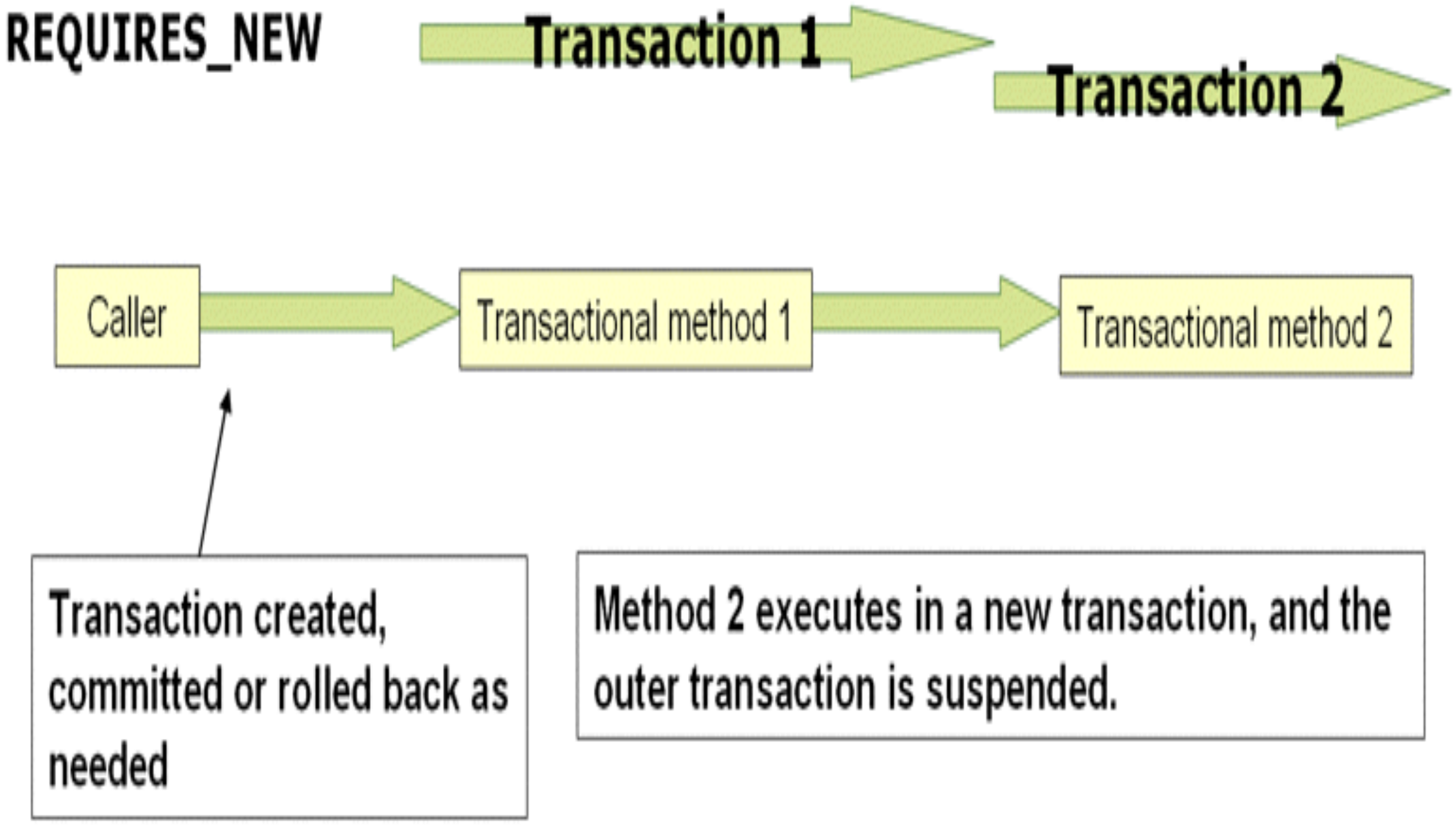
- @Transactional annotation wraps methods with transaction control using transaction manager calls to begin / commit / rollback
  - isolation and propagation level config
  - read-only optimization option
  - timeout (max time before rollback)
  - applied at class or method level

```
@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
        public void updateFoo(Foo foo) {
             // do something
        }
```

Spring Framework: Data Transactions

- Propagation provides JTA information about the type of transaction and how it processes with other transaction
  - PROPAGATION_REQUIRED: DEFAULT, will create a transaction if one is not already created and will participate if one exists.
  - PROPAGATION_REQUIRES_NEW: will create a new transaction and not participate in existing one, if other exists, will suspend it
  - PROPAGATION_SUPPORTS: will use transaction if exists, otherwise will not create one (read only operations)
  - PROPAGATION_NESTED: will leverage save point or sub transactions if supported by resources.
  - Other options that allow for error handling if exception exists or does not


REQUIRED
Transaction
Caller
Transactional method 1
Transactional method 2
Transaction created, committed or rolled back as needed
Method 2 executes in the existing transaction.


REQUIRES_NEW
Transaction 1
Transaction 2
Caller
Transactional method 1
Transactional method 2
Transaction created, committed or rolled back as needed
Method 2 executes in a new transaction, and the outer transaction is suspended.

Spring Framework: Data Transactions

- Unit Testing can utilize a third party Transaction Manager
- Atomikos is open source and very stable
  - Transaction Essentials (Apache License 2)
  - DataSource wrappers for XA and non-XA connections
  - Support for JMS

Spring Framework: Data Transactions

Example XML Config

```
<bean id="dataSource" name="dataSource,datasource"
    class="com.atomikos.jdbc.nonxa.AtomikosNonXADataSourceBean"
    lazy-init="true" init-method="init" destroy-method="close" >
    <property name="uniqueResourceName" value="NONXADBMS" />
    <property name="driverClassName" value="${jdbc.driverClassName}" />
    <property name="url" value="${jdbc.url}" />
    <property name="user" value="${jdbc.username}" />
    <property name="password" value="${jdbc.password}" />
    <property name="readOnly" value="false" />
    <property name="poolSize" value="1" />
    <property name="maxPoolSize" value="4" />
    <property name="minPoolSize" value="0" />
    <property name="testQuery" value="select 1 from dual" />
</bean>

<bean id="atomikosTransactionManager"
    class="com.atomikos.icatch.jta.UserTransactionManager"
    init-method="init" destroy-method="close">
    <property name="forceShutdown" value="true"/>
    <property name="transactionTimeout" value="${transaction.timeout}" />
</bean>
```

Spring Framework: Data Transactions

Example XML Config (cont)

```
  <bean id="atomikosUserTransaction"
      class="com.atomikos.icatch.jta.UserTransactionImp">
    <property name="transactionTimeout" value="${transaction.timeout}" />
  </bean>

<!-- Configure the Spring framework to use JTA transactions from Atomikos --
  >
  <bean id="transactionManager"
      class="org.springframework.transaction.jta.JtaTransactionManager">
    <property name="transactionManager">
        <ref bean="atomikosTransactionManager" />
    </property>
    <property name="userTransaction">
        <ref bean="atomikosUserTransaction" />
    </property>
</bean>


<!-- enable the configuration of transactional behavior based on
annotations -->
<tx:annotation-driven transaction-manager="transactionManager"/>
```

Spring Framework: Data Transactions

- LAB 08 use Spring JDBC Repository to create an AuditDAO implementation
  - Uncomment Asserts in AuditDAOTest
  - Uncomment Asserts in UserTest
  - Change Objects to make test green
  - Use @Transactional appropriately

LAB 08

Wait here until ready to move on…

Spring Framework: Web with Boot

- Provides easy boiler plates for starting apps
  - Support from STS
    File->New->Spring Starter Project
  - http://projects.spring.io/spring-boot

- Easy getting started with InitializR
  - select types of options (Web, JMS, JPA, etc)
  - generates boiler plate zip to download
  - http://start.spring.io

DEMO Spring-Boot

Spring Framework: Resources

http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/
http://spring.io/docs
http://spring.io/guides
http://www.theserverside.com/news/1364527/Introduction-to-the-Spring-Framework
http://www.javabeat.net/tutorials/8-spring-framework-beginners-tutorial.html
Spring In Action, Walls
Pro Spring, Harrop and Machacek

Spring Framework: Resources

http://refcardz.dzone.com/refcardz/spring-configuration
http://refcardz.dzone.com/refcardz/spring-annotations
http://refcardz.dzone.com/refcardz/core-spring-data
http://refcardz.dzone.com/refcardz/eclipse-tools-spring
http://refcardz.dzone.com/refcardz/spring-web-flow
http://refcardz.dzone.com/refcardz/expression-based-authorization
http://refcardz.dzone.com/refcardz/getting-started-with-jpa
http://refcardz.dzone.com/refcardz/whats-new-jpa-20
http://refcardz.dzone.com/refcardz/eclipselink-jpa
http://refcardz.dzone.com/refcardz/spring-integration
http://refcardz.dzone.com/refcardz/spring-batch-refcard

Thank You