

# Spring 2.0 Kickstart

Thomas Risberg  
Matt Raible

 Spring Forward 2006

# Introduction



**Thomas Risberg**



- ▶ Independent Consultant, [springdeveloper.com](http://springdeveloper.com)
- ▶ Committer on the Spring Framework project since 2003
- ▶ Supporting the JDBC and Data Access code
- ▶ Co-author of “Professional Java Development with the Spring Framework” from Wrox



# Sample Application

# Disclaimer!

- This is currently using leading edge and un-released software, things might change slightly and some features might be temporarily broken on occasion.

# How to build the code

**Prerequisites:** *Java SDK 5.0, Maven 2.0.4, Subversion*

## 1. Download source from project at Google Code:

<http://code.google.com/p/spring-kickstart/>

Check-out using Subversion:

```
svn checkout http://spring-kickstart.googlecode.com/svn/trunk/ spring-kickstart
```

## 2. Switch to new project

```
> cd spring-kickstart/kickstart
```

## 3. Run Maven build:

```
> mvn clean package
```

# How to run the code


## 4. Run Maven Jetty plug-in:

```
> mvn jetty:run
```



# Spring 2.0 Kickstart

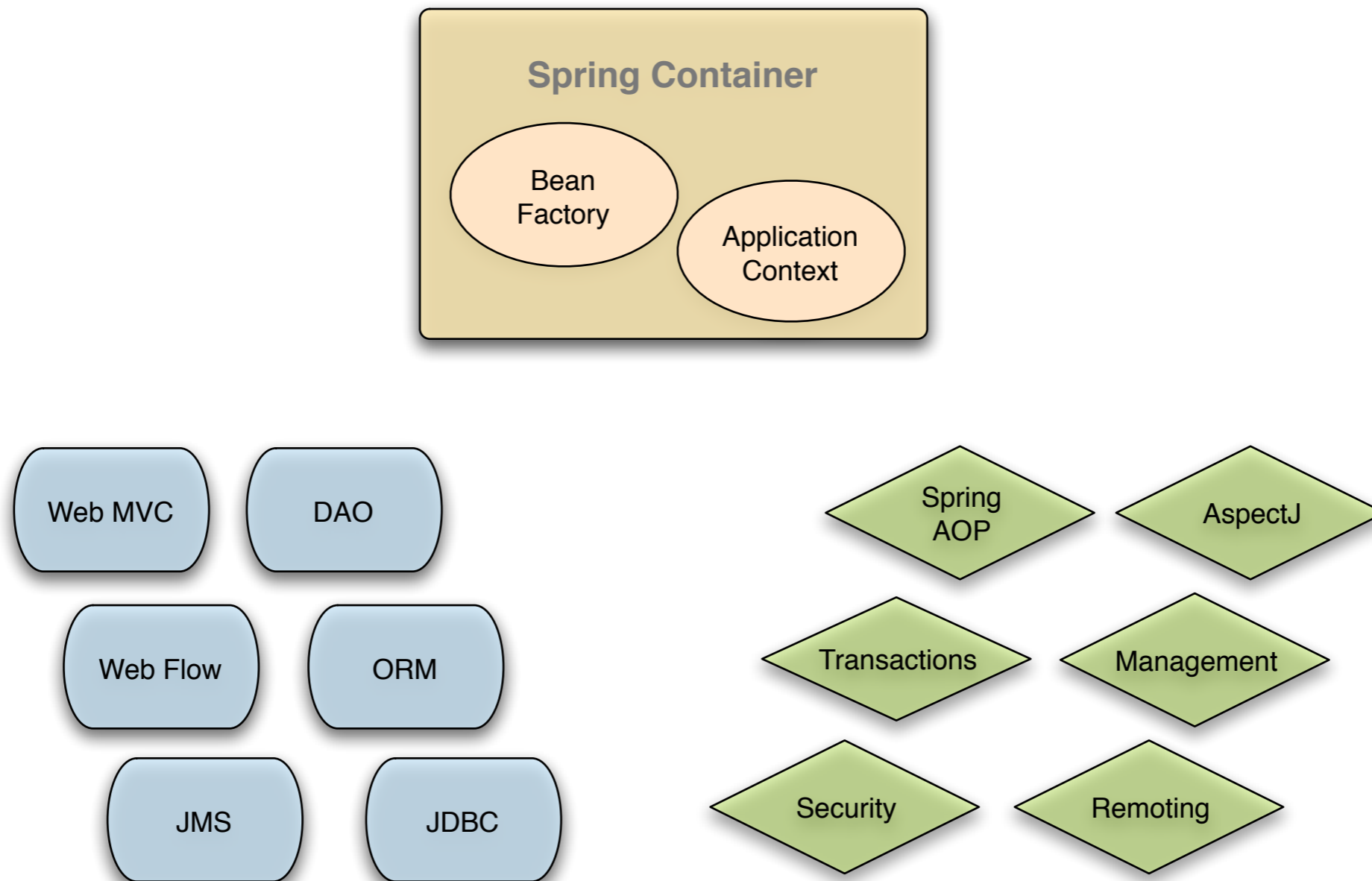
# Introduction

 Spring Forward 2006

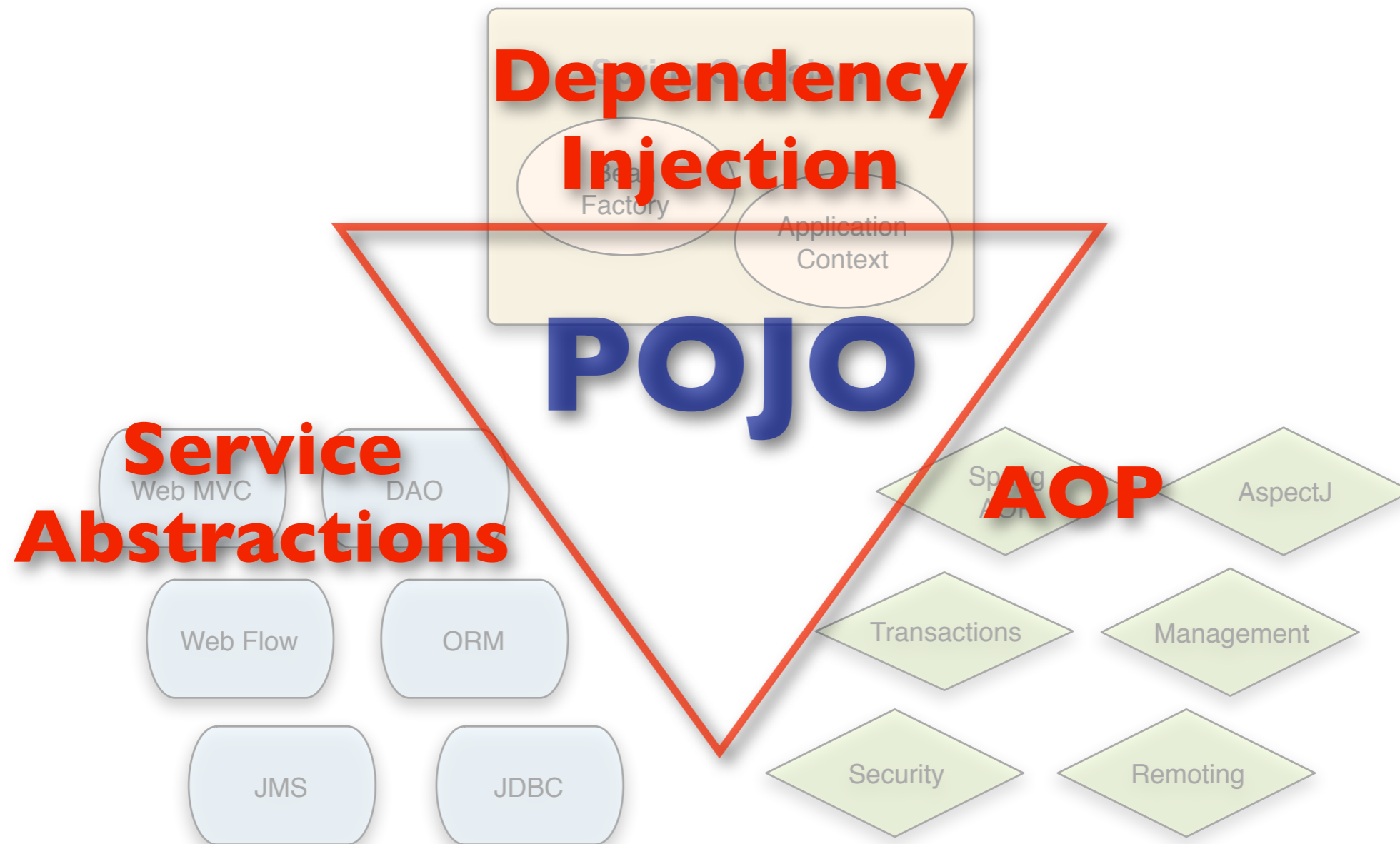
# Intro

- Spring is a Lightweight Application Framework covering all tiers of your typical business application
- Leverages services from underlying runtime environment (e.g. Java EE Services)
- Provides AOP services for security, transactions, management and remoting
- Integrates with other commonly used frameworks and libraries
- Greatly simplifies development effort
- Promotes modular, reusable coding practices

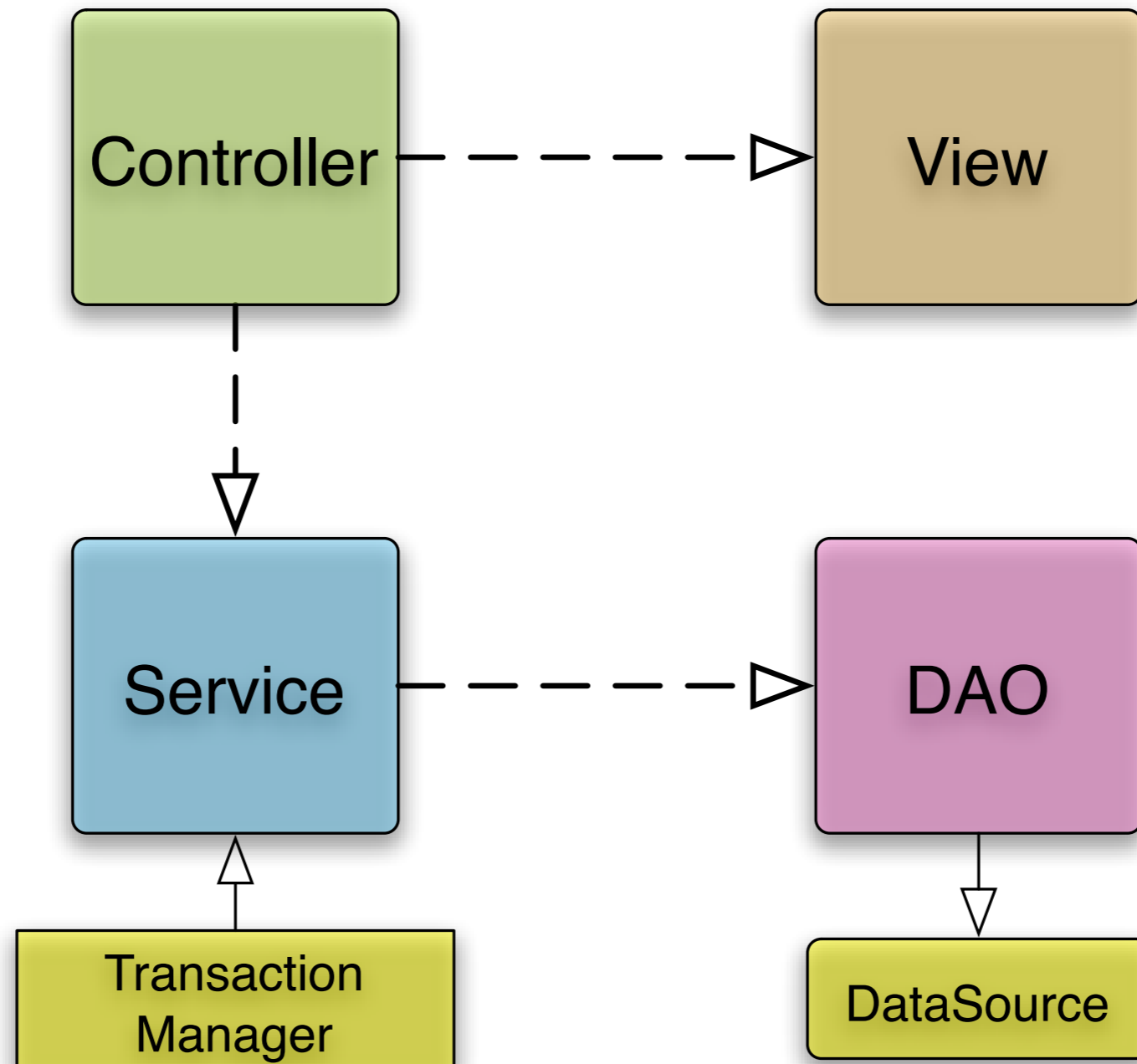
# Spring Triangle



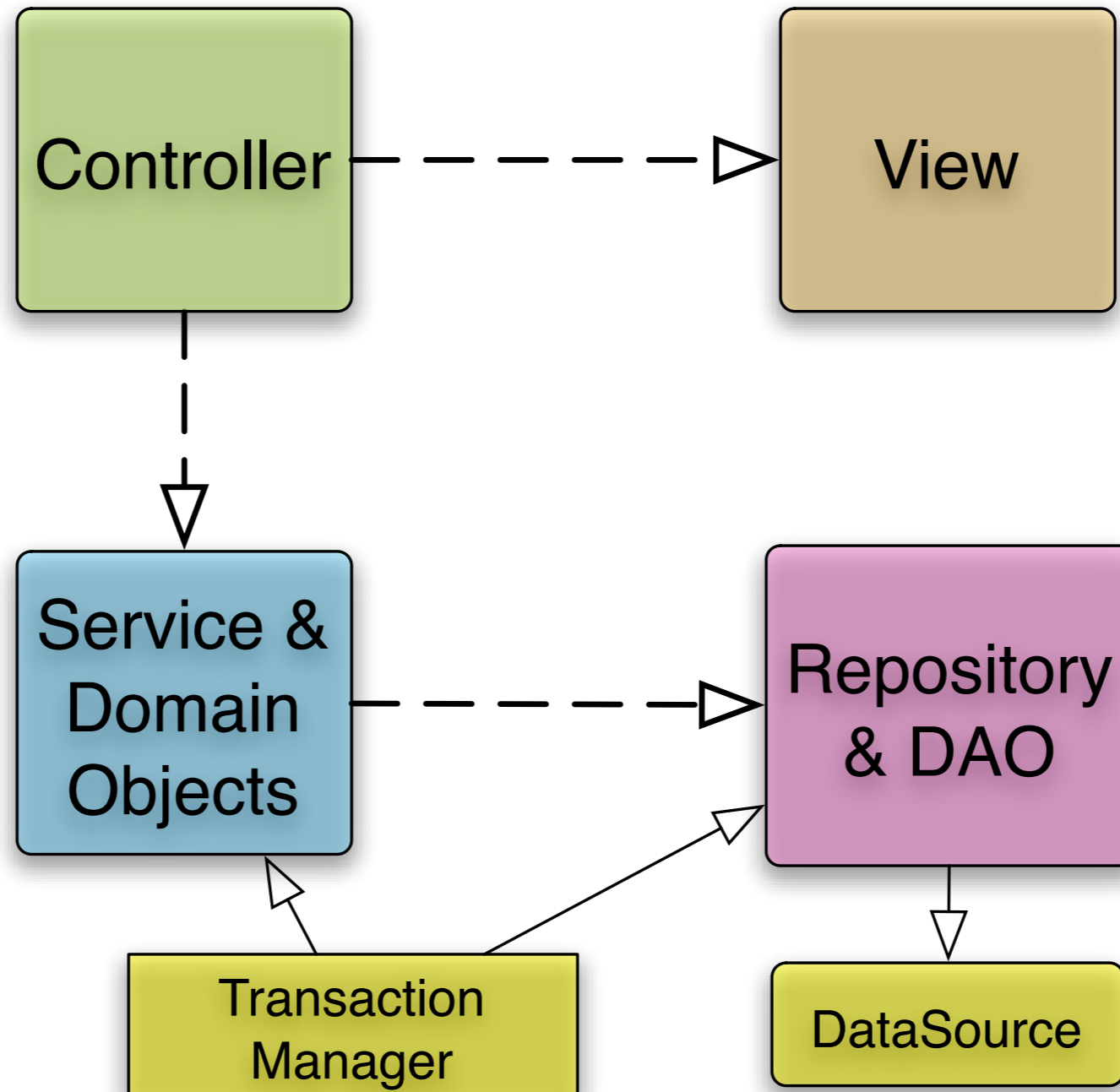
# Spring Triangle



# Typical Spring Web App



# DDD Spring Web App



# Application Context Basics

```
<bean id="widget" class="springdeveloper.bean.Widget">
  <property name="message" ref="message" />
</bean>

<bean id="message" class="java.lang.String">
  <constructor-arg value="Hello, Ruby People" />
</bean>

<bean class="springdeveloper.bean.WidgetPostProcessor" />

<aop:aspectj-autoproxy />
<bean class="springdeveloper.bean.FourLetterWordAspect" />
```

# Application Context Basics

```
public class WidgetPostProcessor implements BeanPostProcessor {
    public Object postProcessBeforeInitialization(Object bean, String beanName)
        throws BeansException {
        if (bean instanceof Widget) {
            System.out.println("Changing: " + beanName);
            if (((Widget)bean).getMessage() != null)
                ((Widget)bean).setMessage(((Widget)bean).getMessage() + "!!!");
        }
        else {
            System.out.println("Skipping: " + beanName);
        }
        return bean;
    }

    public Object postProcessAfterInitialization(Object object, String string)
        throws BeansException {
        return object;
    }
}
```

# Spring 2.0 Kickstart

# Persistence

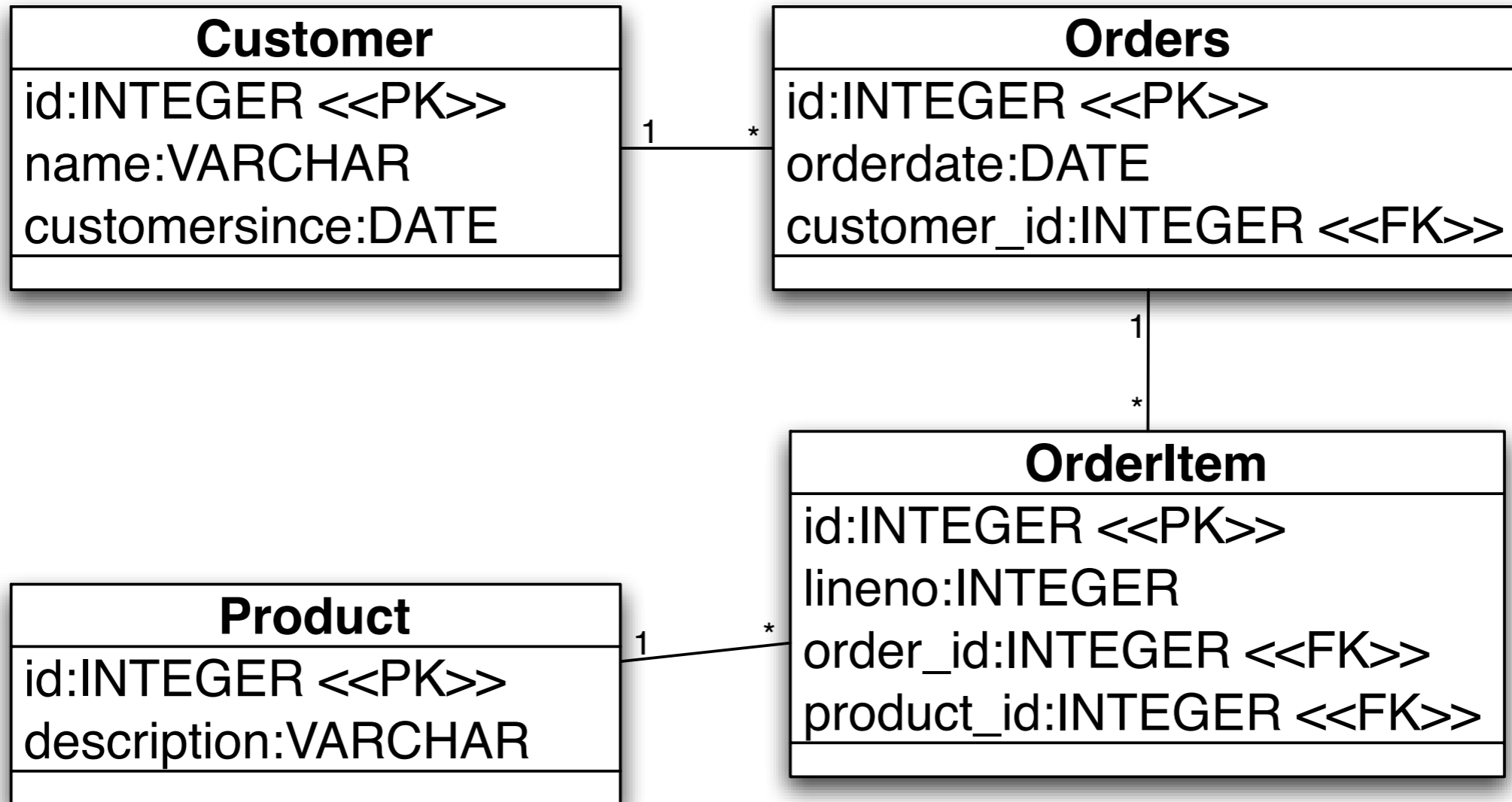
 Spring Forward 2006

# Sample Application

## **Customer Service Application**

- Show a list of customers and choose one for further functionality
- Modify a customer entry
- List all orders for a customer
- Cancel an order that hasn't shipped

# Data Model



# Start new Maven Project

```
mvn archetype:create -DgroupId=spring.kickstart \  
-DartifactId=kickstart \  
-DarchetypeArtifactId=maven-archetype-webapp
```

# Maven pom.xml configuration

```
<build>
  <finalName>kickstart</finalName>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.5</source>
        <target>1.5</target>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.mortbay.jetty</groupId>
      <artifactId>maven-jetty-plugin</artifactId>
      <configuration>
        <scanIntervalSeconds>3</scanIntervalSeconds>
        <contextPath>/</contextPath>
        <scanTargets>
          <scanTarget>src/main/webapp/WEB-INF/kickstart-servlet.xml</scanTarget>
          <scanTarget>src/main/resources/messages.properties</scanTarget>
        </scanTargets>
      </configuration>
    </plugin>
  </plugins>
</build>
```

```
<!-- persistence dependencies -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring</artifactId>
  <version>2.0-rc4</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-mock</artifactId>
  <version>2.0-rc4</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jpa</artifactId>
  <version>2.0-rc4</version>
</dependency>
<dependency>
  <groupId>javax.persistence</groupId>
  <artifactId>persistence-api</artifactId>
  <version>1.0</version>
</dependency>
<dependency>
  <groupId>javax.persistence</groupId>
  <artifactId>toplink-essentials</artifactId>
  <version>1.0</version>
</dependency>
<dependency>
  <groupId>hsqldb</groupId>
  <artifactId>hsqldb</artifactId>
  <version>1.8.0.5</version>
</dependency>
<dependency>
  <groupId>c3p0</groupId>
  <artifactId>c3p0</artifactId>
  <version>0.9.0.4</version>
</dependency>
```

# Customer Class

```
@Entity
public class Customer {
    @Id
    @GeneratedValue(strategy = AUTO)
    private Long id;

    private String name;

    @Temporal(DATE) private Date customerSince;

    @OneToMany(mappedBy = "customer", cascade = ALL)
    Collection<Order> orders;

    public Customer() {}

    public Customer(Long id, String name, Date customerSince) {
        this.id = id;
        this.name = name;
        this.customerSince = customerSince;
    }
}
```

# Order Class

```
@Entity
@Table(name = "orders")
public class Order {
    @Id
    @GeneratedValue(strategy = AUTO)
    private Long id;

    @OneToMany(mappedBy = "order", cascade = ALL)
    @OrderBy("lineNo")
    private List<OrderItem> orderItems;

    @Temporal(DATE) private Date orderDate;

    @ManyToOne
    Customer customer;

    public Long getId() {
        return id;
    }
}
```

# Customer Repository

```
@Repository
@Transactional(propagation = Propagation.SUPPORTS)
public class CustomerRepositoryImpl implements CustomerRepository {

    private EntityManager em;

    @PersistenceContext
    public void setEntityManager(EntityManager entityManager) {
        this.em = entityManager;
    }

    public Customer findById(Long id) {
        return em.find(Customer.class, id);
    }

    public void add(Customer customer) {
        em.persist(customer);
    }

    public List<Customer> findAll() {
        return em.createQuery("select c from Customer c")
            .getResultList();
    }
}
```

# Spring Configuration

- **@Repository**
  - PersistenceExceptionTranslationPostProcessor
- **@Transactional**
  - `<tx:annotation-driven/>`
  - JpaTransactionManager
- **@PersistenceContext**
  - LocalContainerEntityManagerFactoryBean
  - PersistenceAnnotationBeanPostProcessor

# repository-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:tx="http://www.springframework.org/schema/tx"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx-2.0.xsd">

  <bean id="customerRepository" class="spring.kickstart.repository.CustomerRepositoryImpl"/>
  <bean id="productRepository" class="spring.kickstart.repository.ProductRepositoryImpl"/>
  <bean class="org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor"/>
  <bean class="org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor"/>

  <bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory"
      ref="entityManagerFactory" />
  </bean>

  <tx:annotation-driven />
  ...
```

# repository-config.xml

```
...  
<bean id="entityManagerFactory"  
  class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">  
  <property name="dataSource" ref="dataSource"/>  
  <property name="persistenceUnitName" value="kickstart"/>  
  <property name="jpaVendorAdapter">  
    <bean class="org.springframework.orm.jpa.vendor.TopLinkJpaVendorAdapter">  
      <property name="showSql" value="false"/>  
      <property name="generateDdl" value="false"/>  
      <property name="databasePlatform" value="Derby"/>  
    </bean>  
  </property>  
  <property name="loadTimeWeaver">  
    <bean class="org.springframework.instrument.classloading.SimpleLoadTimeWeaver"/>  
  </property>  
</bean>  
...
```

# repository-config.xml

....

```
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
    destroy-method="close">
    <property name="driverClass" value="${db.driverClassName}"/>
    <property name="jdbcUrl" value="${db.url}"/>
    <property name="user" value="${db.userName}"/>
    <property name="password" value="${db.password}"/>
    <property name="minPoolSize" value="2"/>
    <property name="maxPoolSize" value="15"/>
    <property name="maxStatements" value="50"/>
</bean>

<bean id="propertyConfigurer"
    class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="locations">
        <list>
            <value>jdbc.properties</value>
        </list>
    </property>
</bean>

</beans>
```

# CustomerRepositoryTest

```
public void testFindCustomer() {
    Customer c = customerRepository.findById(testId);
    assertEquals(testId, c.getId());
    assertEquals("Test", c.getName());
}

protected String[] getConfigLocations() {
    return new String[] {"repository-test-config.xml"};
}

@Override
protected void setUpInTransaction() throws Exception {
    EntityManager em =
        EntityManagerFactoryUtils.getTransactionEntityManager(emf);
    Customer c = new Customer();
    c.setName("Test");
    c.setCustomerSince(new Date());
    em.persist(c);
    testId = c.getId();
    super.setUpInTransaction();
}
```

# CustomerRepositoryTest

```
protected void onSetUpInTransaction() throws Exception {

    EntityManager em =
        EntityManagerFactoryUtils.getTransactionalEntityManager(emf);

    Product p1 = new Product();
    p1.setDescription("Product1");
    em.persist(p1);

    Customer c = new Customer();
    c.setName("Test");
    c.setCustomerSince(new Date());
    Order ol = new Order();
    OrderItem oil = new OrderItem();
    oil.setLineNo(1);
    oil.setOrder(ol);
    oil.setProduct(p1);
    List ois = new ArrayList();
    ois.add(oil);
    ol.setOrderItems(ois);
    List os = new ArrayList();
    os.add(ol);
    c.setOrders(os);

    em.persist(c);

    super.onSetUpInTransaction();
}
```

Q & A

# Persistence

***Don't miss JPA talks this afternoon!***