JDBC and Web Development with the Spring Framework

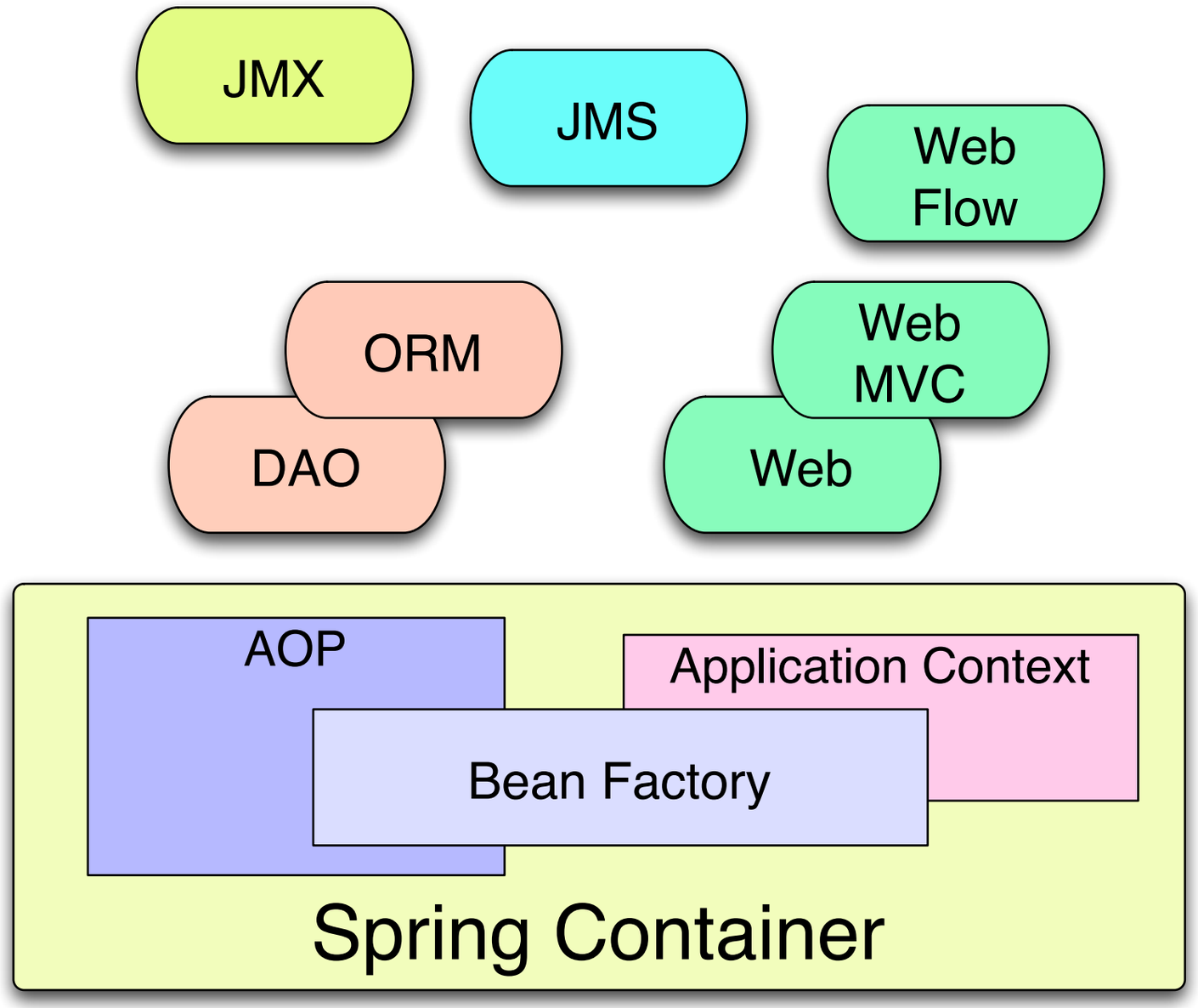
Thomas Risberg
Rob Harrop

OSCON 2005

Outline

- Introduction to Spring and Dependency Injection
- Spring's JDBC Framework
- Web Development with Spring MVC
- Transactions
- Integration and the Open Source Project

Introducing Spring



Spring Dependency Injection Basics

- ✓ Beans
- ✓ BeanFactory
- ✓ Configuration file formats
- ✓ Simple example of DI in action

Simple Example

```
public class TestBean {
     private String name;

     public void setName(String name) {
          this.name = name;
     }

     public String getName() {
          return name;
     }
}
```

```
public class TestClass {
     private String name;

     public TestClass(String name) {
          this.name = name;
     }

     public String getName() {
          return name;
     }

}
```

Simple Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
        "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

    <bean id="myBean" class="TestBean">
        <property name="name" value=”Thomas”/>
    </bean>

    <bean id="myClass" class="TestClass">
        <constructor-arg ref="aName"/>
    </bean>

    <bean id="aName" class="java.lang.String">
        <constructor-arg value=”Rob”/>
    </bean>

</beans>
```

Simple Example

```
public class DISample {
    private ApplicationContext ac;

    public static void main(String[] args) {
        DISample di = new DISample();
        di.run();
    }

     public void run() {
        ac = new FileSystemXmlApplicationContext("src/applicationContext.xml");
        TestBean tb = (TestBean) ac.getBean("myBean");
        TestClass tc = (TestClass) ac.getBean("myClass");
        System.out.println("MyBean: " + tb.getName());
        System.out.println("MyClass: " + tc.getName());
    }
}
```

```
MyBean: Thomas
MyClass: Rob
```

Dependency Injection Advantages

- Avoid adding lookup code in business logic
- Promotes a consistent approach across all applications and teams
- Simplifies unit testing
- Allows reuse in different application environments by changing configuration files instead of code

Structure of Typical Spring Web Application