



# Spring Framework Introduction

Khader Shaik

# Agenda

---

- Background
- J2EE Vs Spring
- Framework Mission
- Modules
- Application Development Process
- Sample Application

# Spring Framework

- Spring is a Java Enterprise Application Framework
  - Lightweight Application Framework
  - Support all layers including web layer
  - Provides all foundation components required to build the enterprise application
  - Open source framework
- Spring 1.0 was released in 2004
- Initially developed by Rod Johnson as part of the book “J2EE Design & Application” in 2002/2003
- Spring.Net in pipeline for .Net platform
- [www.springframework.org](http://www.springframework.org)

# Spring Framework

---

- Spring is not a J2EE Application Server
- Spring works in simple Java environment
- Spring can be integrated with any Application Server
- Spring simplifies J2EE development
- Supports POJO model
  - POJO – Plain Old Java Objects

# Other Frameworks - Struts

---

- There are many Java application Frameworks like Struts etc
- Struts designed for Web Layer
- Other Frameworks also addressed specific layer
- But Spring Framework provides solution to support all layers of application

# J2EE Vs Spring

---

- J2EE
  - Enterprise Architecture with large scope
  - Heavyweight Architecture
  - Run in J2EE Container - dependent
  - Complex to implement and test
  - Many J2EE APIs are not modular (tightly integrated)
  - Difficult to develop simple and small applications

# J2EE Vs Spring cont..

---

- Spring
  - Spring simplifies the J2EE application development
  - Spring doesn't need J2EE container
  - Supports modular services and standard integration
  - Easy to implement and test the code
  - Spring core is Based on Pattern called "Inversion of Control" (IoC)
  - Open Source Framework
  - Quick development
  - Easy to implement small and simple applications
  - Spring allows to build reusable business and data objects that do not depend on J2EE services

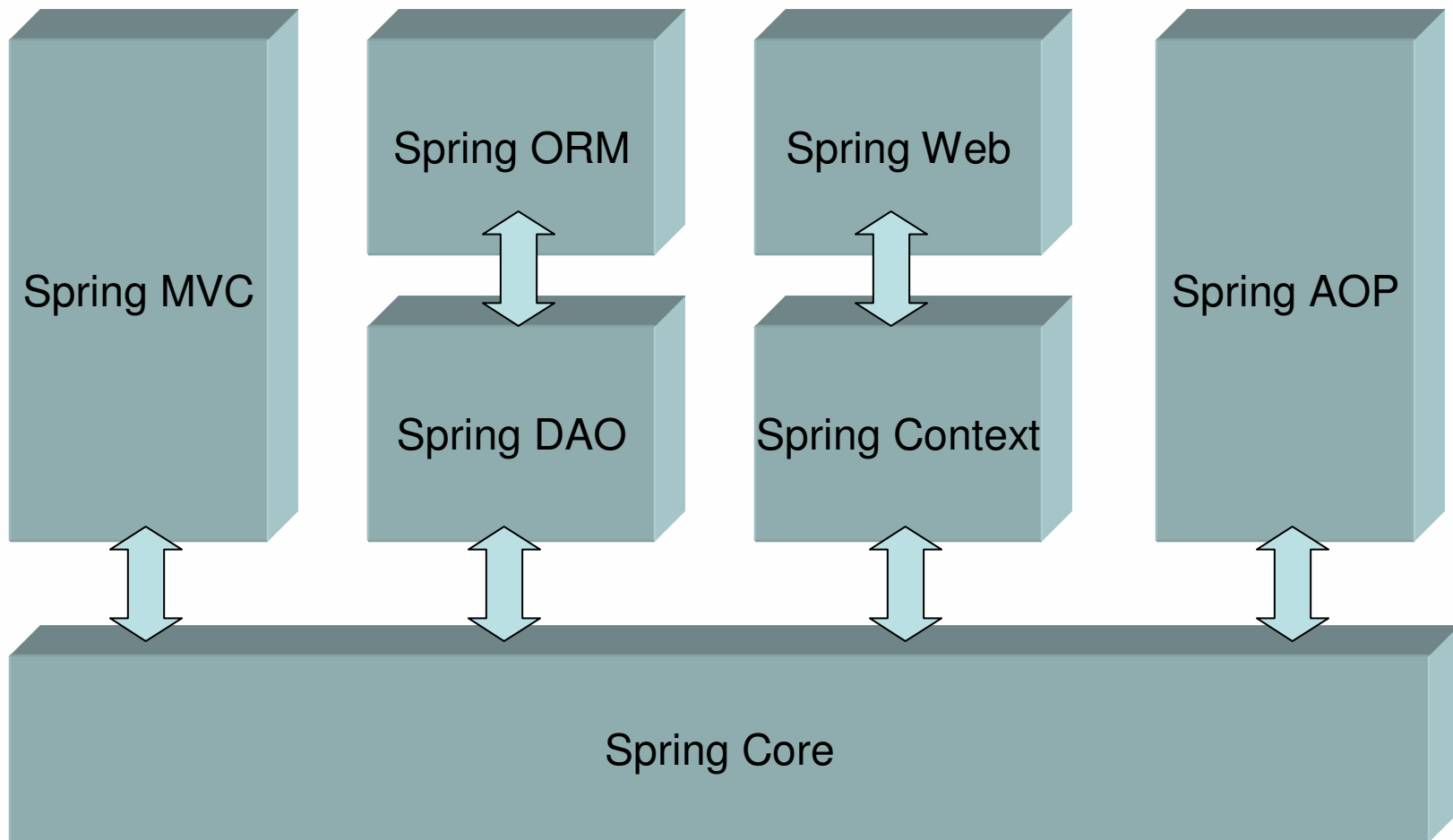
# Spring Framework Mission

- J2EE should be easier to use
- It's best to program to interfaces, rather than classes. Spring reduces the complexity cost of using interfaces to zero.
- JavaBeans offer a great way of configuring applications.
- OO design is more important than any implementation technology, such as J2EE.
- Checked exceptions are overused in Java. A framework shouldn't force you to catch exceptions you're unlikely to be able to recover from.
- Testability is essential, and a framework such as Spring should help make your code easier to test.

- *From Spring Authors*



# Spring Framework Modules



# Spring Modules

---

- Spring is combination of various modules
  - Seven well defined modules
  - Most of them are reasonably independent
- Spring modules built using modular approach
  - You can use only required modules
  - Each module is set of one or more JAR files
- Spring's core module is "Inversion of Control" (IoC) also known as "Dependency Injection"
- All other Spring modules are built on top of IoC
- IoC is a Foundation or Container of Spring Framework

# Spring Other Modules

---

- Spring AOP
- Spring DAO
- Spring ORM
- Spring Context
- Spring Web
- Spring Web MVC

# IoC

- Core Container that provides the essential functionality
- It's a Primary Component and a Foundation
- IoC allows the different Application model
  - Describes the application design contract (interfaces)
  - Contract explains – what it does and how it behaves
  - Doesn't enforce how to create actual objects
- Main part of the IoC is BeanFactory
  - It implements Factory Pattern
- IoC creates objects and wires them based on application description
  - Links methods to be invoked with application description
- Object (POJO) functionality and behavior is described using its configuration files (XML)
- BeanFactory is the core component in IoC
- XmlBeanFactory loads beans (POJOs) based on definitions in a XML file

# BeanFactory Interface

---

- Commonly used Implementation – XMLBeanFactory
- Implements Factory Method Pattern
- XMLBeanFactory loads Beans based on the definition in XML file
- Available in the package – *org.springframework.beans*

```
BeanFactory bFactory = new XMLBeanFactory(  
    new FileInputStream("SecMasterBeans.xml"));  
SecurityBean secBean = (SecurityBean) bFactory.getBean("securitybean");
```

# Spring Context

---

- It is a configuration file that provides context information to the Spring Framework
- Used to provide enterprise services like
  - JNDI
  - EJB
  - Internationalization etc

# Spring AOP

---

- Provides the Aspect Oriented Programming functionality in Spring Framework
- AOP provides transaction management services for Spring Objects
- Supports declarative transaction management
- AOP (out of scope)

# Spring DAO

---

- Provides better exception handling and error reporting from Database sources
- Helps to reduce the amount of Exception handling code



# Spring ORM

---

- ORM – Object Relational Mapping
- Spring ORM supports various ORM frameworks like JDO, Hibernate, iBatis SQL Maps etc
- Integrated with Spring DAO for better exception handling and error messaging

# Spring Web

---

- Provides the Context for the Web based applications
- Built on top of Spring Context module
- Supports the integration with other popular Web frameworks like STRUTS

# Spring MVC

---

- Spring MVC is the implementation of Model-View-Controller architecture
- It is highly configurable and full implementation
- Supports many different pluggable components as part of View layer
  - Velocity
  - JSP
  - Tiles
  - POI
  - iText etc

# Spring Application Development

---

- Application Run-time requirements
  - *Java Runtime*
  - *Spring Framework Libraries*
  - *Other Application Libraries that your application uses*
  - *Your Application Object Libraries*
- *Development Environment Requirements*
  - *JDK 1.4.2 or later*
  - *Spring Framework*
  - *Other Application Libraries that your application uses*
  - *IDE*
  - *Ant or other make utility*

# Application Development Steps

---

- *Business Components*
  - *Develop Business Logic Beans – POJOS*
  - *Develop the Configuration files to map the Business Objects*
- *Client*
  - *Develop the Client Classes*

# Simple Application

---

- *Consider application that is using spring framework*
- *Container loads the Customer Bean as per the client's request and runs the method on it*
- *Components*
  - *Customer Bean (customer info)*
  - *Mapping File – instructs the IoC on where load the Bean from*
  - *Client – instantiates the Spring container, loads the business logic bean and executes the method*

# Simple Bean – Customer Class

```
public class Customer {
    private String id;
    public String getId() {
        return name;
    }
    public void setId(String name) {
        this.id = id
    }
    public long accountBalance(String id) {
        // retrieve the balance
    }
}
```

# Configuration/Mapping – beans.xml

```
<beans>
  <bean
    id="customerbean"
    class="com.orbita.Customer"
    autowire="byType">
    <property name="id"/>
  </bean>
  ....
</beans>
```



# Client Class

---

```
public class TestClient {  
    ...  
    public static void main(String args[]){  
        BeanFactory bFactory = new XMLBeanFactory(  
            new FileInputStream("beans.xml"));  
        Customer cust = (Customer) bFactory.getBean("customerbean");  
        System.out.println("balance: " + cust.getBalance("JOHN432"));  
    }  
}
```

*Thank You*  
*Khader Shaik*  
[khaderv@yahoo.com](mailto:khaderv@yahoo.com)