

ORACLE®



ORACLE®

Spring in WebLogic und SOA Suite

Marcel Amende

Leitender Systemberater - Business Unit Middleware

Oracle Deutschland B.V. & Co. KG

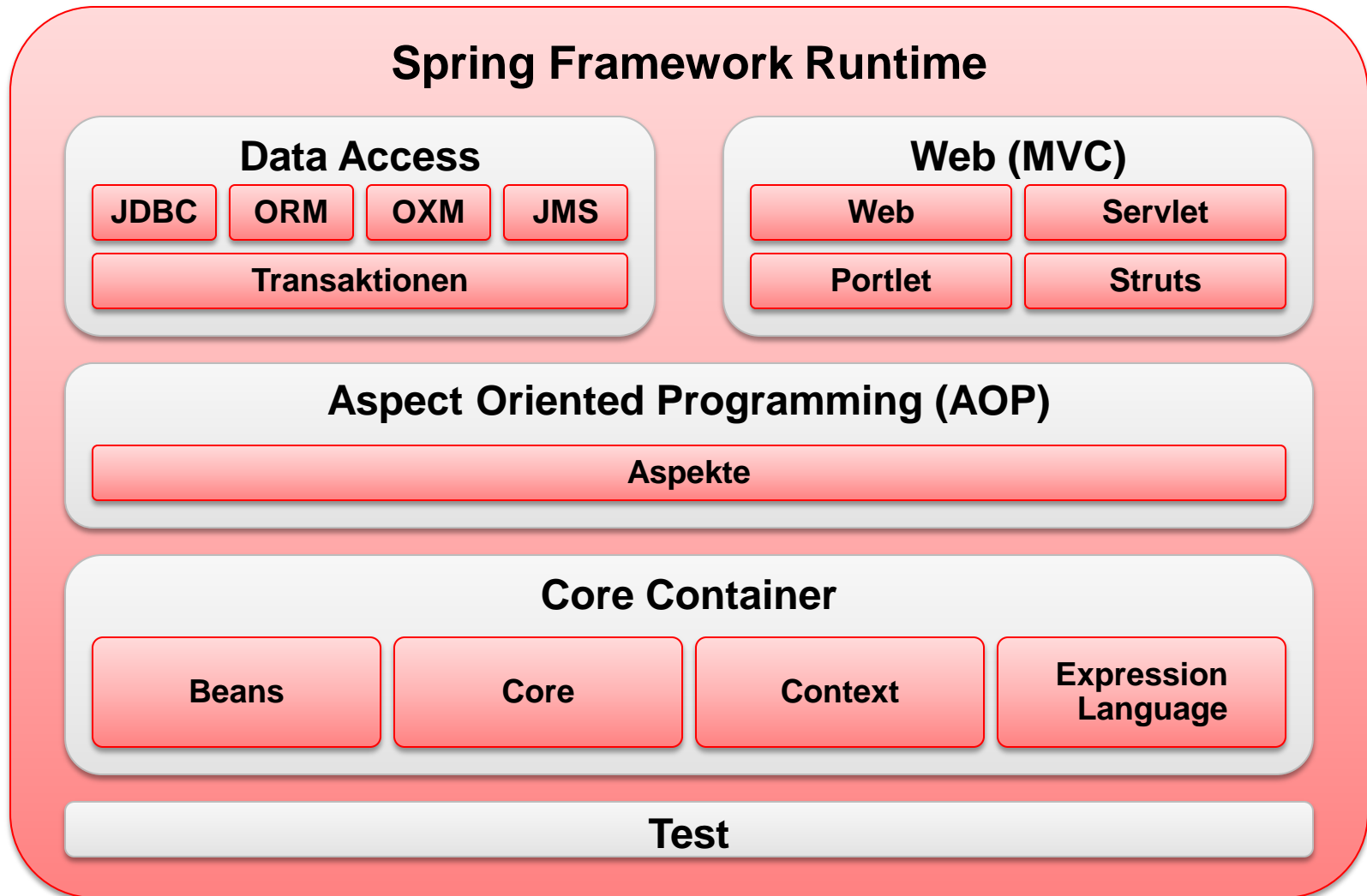
Agenda

- Spring Framework
- Spring Module
- Spring im WebLogic Server
- Spring und SCA
- Spring in der SOA Suite

Spring Framework

- open source Framework für die Java Plattform
- basiert auf der Publikation „Expert One-to-One J2EE Design and Development“ von Rod Johnson
- Ziele:
 - Vereinfachung der JEE-Nutzung
 - Entkopplung durch vereinfachte Nutzung von Schnittstellen
 - Konfiguration über Java Beans
 - Konzentration auf Objektorientierung, nicht auf Technologie
 - bessere Testbarkeit
- Alternative/Ergänzung zum (komplexen) EJB-Modell
- modularer Aufbau

Spring Module



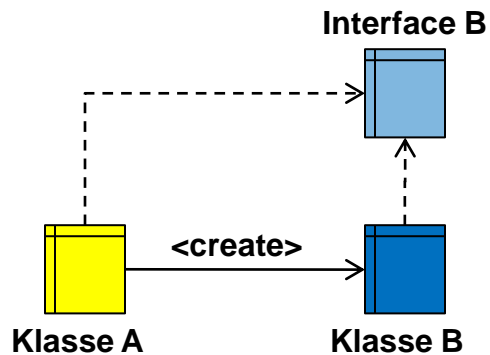
Spring Module

„Inversion of Control (IoC)“ Container

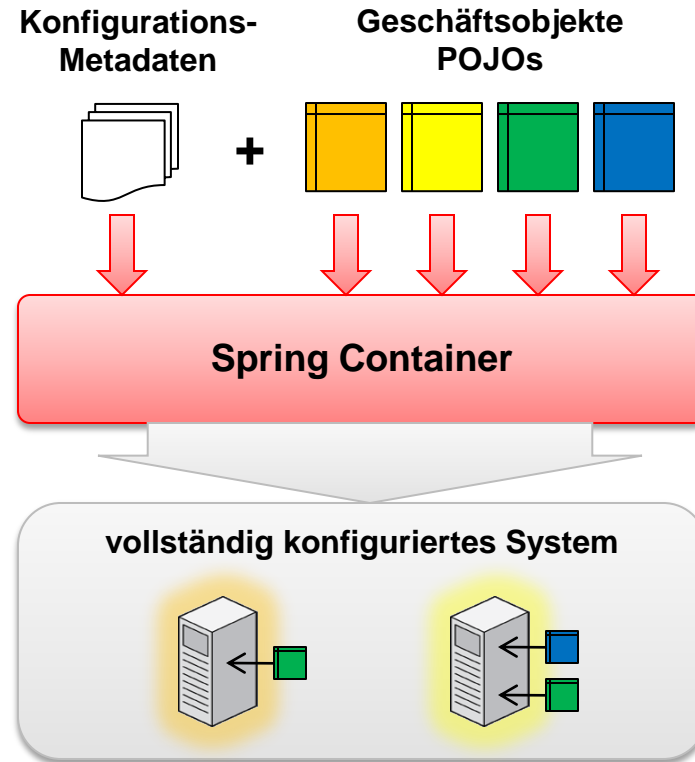
- Container verantwortet Lebenszyklus von Objekten
- Umkehrung der prozeduralen Programmierung:
 - prozedural: zentraler Code kontrolliert die Geschäftslogik und ruft spezifische Subroutinen auf
 - IoC: generischer Code kontrolliert die Ausführung von unabhängigem, problemspezifischem Code
- „Dependency Injection“:
 - Injektion benötigter Komponenten von aussen
 - statt direkter Kopplung im Code oder Aufruf eines Dienstes
 - Klasse muss nicht wissen, *wie* sie etwas bekommt
 - Entkopplung der Klasse von der Laufzeitumgebung
 - alternativ: „Dependency Lookup“: Frage nach best. Klasse

„Inversion of Control (IoC)“

klassisches System



Umkehr der Kontrolle mit „Dependency Injection“



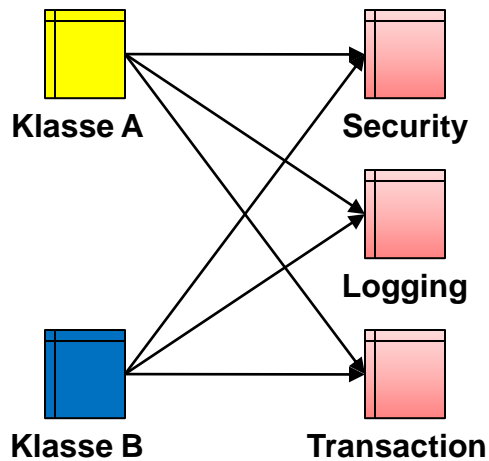
Spring Module

„Aspect Oriented Programming“ Framework

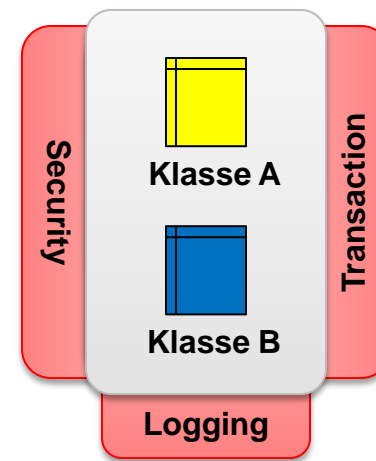
- Modularisierung von Querschnittsthemen in Aspekte: Transaktionsmanagement, Sicherheit, entfernte Zugriffe, JMX
- deklarative Anreicherung von containerkontrollierten Objekten
- Abfangroutinen, zur Laufzeit konfiguriert

„Aspect Orientierted Programming (AOP)“

ohne AOP



mit AOP

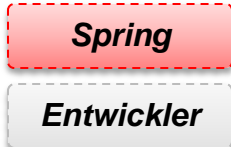
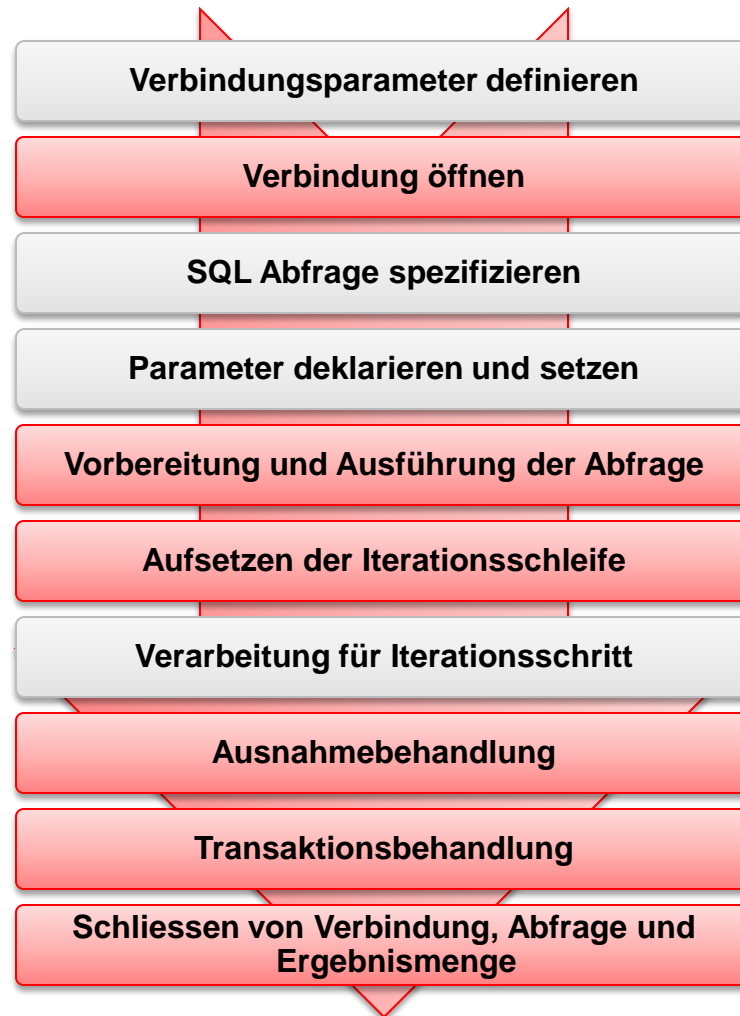


Spring Module

„Data Access“ Framework

- Unterstützung für JDBC, JPA, TopLink, Hibernate, ...
- Nutzung über Vorlageklassen („Templates“)
 - automatisches Ressourcenmanagement
 - einheitliche Fehlerbehandlung
 - vereinfachtes Sessionmanagement
 - integrierte Transaktionsverwaltung

JDBC Vorlage: Wer macht was?



Spring im WebLogic Server

- Unterstützung für „open source“ Spring-Projekte
- Version 2.5.(3+) in WebLogic 10.1.3+
- Spring Console, basierend auf MBeans
- vererbte Eigenschaften:
 - Clustering, Cluster Deployment, Session Replication, Clustered Remoting (RMI Interface für POJOs)
- Integration mit WebLogic Server:
 - Transaction Manager: verteilte Transaktionen
 - JMS: Message Driven POJOs
 - JMX
 - Security

Spring Integration mit Weblogic Security

- „WebLogic Security“:
Erweiterung der „Java EE Security“ durch anpassbare „Security Provider“
- „Spring Security“:
offizielles Sicherheitsprojekt des Spring Portfolios
- kombinierte JEE und Spring Applikation:
 - „WLS Security“: Authentifizierung über „Default Provider“ für den „Security Realm“
 - + Mapper-Klasse konvertiert „WLS-“ in „Spring GrantedAuthority“ Principals für auth. nach „Spring Security“
 - = beiderseitig können nun Ressourcen mit dem jeweiligen Mechanismus gesichert werden

Spring Interaction mit der „Service Component Architecture (SCA)“

- Spring Beans als Servicekomponente
- „Services“ zur Bereitstellung von Komponenteneten als Web Service oder EJB(3)
- „Referenzen“ zum Aufruf von weiteren Servicekomponenten und Diensten
- SCA Runtime im WebLogic verfügbar (shared library)
- Vorgehensweise:
 - „POJO“ bildet Geschäftslogik ab
 - „Spring Context“ konfiguriert die POJOs und deklariert die SCA Services und Referenzen
 - „weblogic.xml“ importiert die SCA shared library

BPEL Process Manager 10g

Java Integration I

- „Enterprise Java Bean“ (EJB):
 - komplexe Spezifikation
 - Bean-Klasse, Home- und Remote-Interfaces nötig
 - eigenes Serverdeployment des Beans + EJB Client nötig
- BPEL Aktivität: „Java Embedding“:
 - proprietäre Erweiterung des BPEL Standards (`<bpelx:exec>`), Bearbeitung des BPEL-Codes nötig
 - nur für kleine Java Code-Fragmente geeignet („Inline Code“)
 - eingeschränkte Möglichkeiten bei Bearbeitung (z.B. keine „Type-In“-Unterstützung) und Fehlersuche
 - aufwändiger Zugriff auf BPEL Variablen (per Hand auf einzelne Nodes oder über zu generierende XML-Façaden)

BPEL Process Manager 10g

Java Integration II

- „Web Service Invokation Framework“ (WSIF):
 - Service-Binding für Java-Klassen und EJBs
 - Beschreibung durch WSDL
 - selbst zu erstellen:
 - XML-Façade für Java-Klasse
 - WSDL mit Java-/EJB-Binding
 - Konfliktpotential mit dem BPEL Class-Loader

Spring Integration mit der SOA Suite 11g

The screenshot displays the Oracle JDeveloper 11g IDE interface for a Spring Integration composite application named "SpringFileTest".

Project Structure: The left sidebar shows the project hierarchy, including source files like `Unzipper.java`, `UnzipperIntf.java`, `ZipFile.java`, and various WSDL and JCA files.

Design View: The central canvas shows a flow diagram for the composite. It starts with a `ReadFilenameService` component, followed by an `UnzipBPELP...` component, then an `UnzipSpring` component, and finally two parallel `GetFileContent` and `CopyFileService` components.

Source Code: The bottom window shows the XML configuration for the beans and service in `UnzipSpring.xml`:

```
<?xml version="1.0" encoding="windows-1252" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:util="http://www.springframework.org/schema/util"
       xmlns:jee="http://www.springframework.org/schema/jee"
       xmlns:lang="http://www.springframework.org/schema/lang"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:sca="http://xmlns.oracle.com/weblogic/weblogic-sca"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spr
<!--Spring Bean definitions go here-->
<bean name="unzipper" class="my.Unzipper"/>
<sca:service name="UnzipperService" target="unzipper" type="my.UnzipperIntf"/>
</beans>
```

Fragen & Antworten

