Spring Framework

Überblick

- Geschichte
- Basics
  - Lightweight Container
  - Inversion of Control (IoC)
  - Dependency Injection (DI)
  - Beans
- Spring Framework
  - Module
- Referenzen

Geschichte

- Grundlage: Rod Johnson, „Expert One-on-One: J2EE Design and Development“
- Komplexität von JEE-Projekten reduzieren
- Modularer Aufbau -> „Cherry Picking“
- Unabhängig von JEE, aber gute Integration mit verschiedenen JEE-Technologien

Basics: Lightweight Container

- Keine bzw. minimale Abhängigkeit zu Container API
- Basiert auf Plain Old Java Objects (POJOs)
- Kurze Turn-around-Zeiten in der Entwicklung
- Kein komplexes Deployment
- Problemlose Integration in andere Umgebungen

Basics: Inversion of Control (IoC)

- „Hollywood principle“, „Don‘t call us, we‘ll call you.“
- Implementierung von Framework-Callback-Methoden
- Beispiele im JDK:
  - ActionListener Interface in Swing
  - Runnable Interface

Basics: Dependency Injection (DI)

- Spezielle Anwendung von IoC
- Erzeugung und Initialisierung von Objekten inklusive deren Abhängigkeiten zu anderen Objekten
- Ctor / Setter Injection

Basics: Beans

- POJOs, mittels Konfiguration zusammengesetzt
- Grundsätzlich keine Beziehung zum Spring-Framework
- Evtl. deklarativ mit zusätzlichen Diensten wie Transaktionssteuerung angereichert
- Instanzierung via Ctor / Factory
- Early / Lazy Initialisierung

Spring Framework: Module I

- Core
  - IoC Container / Dependency Injection
  - Resourcen
  - Validierung
  - AOP
  - Testing

IoC Container / DI I

```
…
public interface PizzaOrderService
{
   public Integer createOrder (Integer aPizzaId);
}
…
public class PizzaOrderServiceImpl implements PizzaOrderService
{
   private PizzaDAO m_PizzaDAO = null;

   public Integer createOrder (Integer aPizzaId)
   {
      System.out.println ("Pizza [" + m_PizzaDAO.getPizzaName (aPizzaId) + "] bestellt!");
      return 1000 + aPizzaId;
   }

   public void setPizzaDAO (PizzaDAO pizzaDAO)
   {
      m_PizzaDAO = pizzaDAO;
   }

}

…
```

IoC Container / DI II

```
…
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
     http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">
   <bean id="pizzaDAO" class="de.opiz.service.PizzaDAOImpl"/>
   <bean id="pizzaOrderService" class="de.opiz.service.PizzaOrderServiceImpl">
      <!-- ctor dependency injection -->
      <constructor-arg>
         <ref bean="pizzaDAO"></ref>
      </constructor-arg>
      <!-- end ctor dependency injection -->
      <!-- setter dependency injection -->
      <!--
      <property name="pizzaDAO">
         <ref bean="pizzaDAO"></ref>
      </property>
         -->
      <!-- end setter dependency injection -->
   </bean>
</beans>
…
```

IoC Container / DI III

```
…
package de.opiz.service;

import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import junit.framework.TestCase;

public class SimpleTest extends TestCase
{

   public void testPizzaService () throws Exception
   {
      AbstractApplicationContext ctx =  new ClassPathXmlApplicationContext ("application-context.xml");
      ctx.registerShutdownHook ();
      PizzaOrderService myService = (PizzaOrderService) ctx.getBean ("pizzaOrderService");

      assertNotNull (myService);

      assertEquals (new Integer ("1001"), myService.createOrder (new Integer ("1")));
   }
}
…
```

Resourcen I

- Abstraktion von Zugriff auf low-level Resourcen durch Resource Interface

```
public interface Resource extends InputStreamSource
{
   boolean exists();
   boolean isOpen();
   URL getURL() throws IOException;
   File getFile() throws IOException;
   Resource createRelative(String relativePath) throws IOException;
   String getFilename();
   String getDescription();
}
```

Resourcen II

- Implementierungen
  - UrlResource
  - ClassPathResource

```
AbstractApplicationContext ctx =  new
  ClassPathXmlApplicationContext ("application-context.xml");
```

  - FileSystemResource
  - ServletContextResource
  - InputStreamResource
  - ByteArrayResource

Testing

- Unterstützung für Mock Objects
- Generische Testinfrastruktur
- Abstrakte Hilfsklassen für JUnit, TestNG

Spring Framework: Module II

Middle Tier

- Transaction Management
- DAO
- JDBC
- ORM

Transaction Management

- Abstraktion von Transaction APIs wie JTA, JDBC, Hibernate, JPA, JDO
- Deklaratives Transaktionsmanagement (XML/Annotations, AOP)
- Einfache API für programmatisches Transaktionsmanagement
- Integration mit Datenzugriffs-Abstraktionen von Spring
- Transaktionsmanager durch Konfigurationsänderung austauschbar

DAO

- Abstraktion von Datenzugriffstechnologien wie JDBC, Hibernate, JDO

JDBC I

- Abstraktion von low-level Details
- Resourcen-Management (Statements / ResultSets schliessen)

JDBC II

```
…
Collection actors = this.jdbcTemplate.query(
   "select first_name, surname from t_actor",
   new RowMapper()
   {
        public Object mapRow(ResultSet rs, int rowNum) throws
   SQLException
        {
                Actor actor = new Actor();
                actor.setFirstName(rs.getString("first_name"));
                actor.setSurname(rs.getString("surname"));
                return actor;
        }
   });
…
```

JDBC III

```
…

public int countOfActorsByFirstName(String firstName)
{
    String sql = "select count(0) from T_ACTOR where first_name = :first_name";
    SqlParameterSource namedParameters = new MapSqlParameterSource("first_name",
    firstName);
    return namedParameterJdbcTemplate.queryForInt(sql, namedParameters);
}
…
```

ORM

- Integration mit verschiedenen ORM Frameworks:
  - Hibernate
  - JDO
  - Oracle TopLink
  - iBatis SQL Maps
  - JPA

Spring Framework: Module III

Web

- Web MVC Framework
- Integration von View Technologien
  - JSP / JSTL
  - Tiles
  - Velocity / Freemarker
  - XSLT
  - Document Views (Excel / PDF)
  - JasperReports
- Integration von anderen Web Frameworks
  - JSF
  - Struts
  - Tapestry
  - WebWork
- Portlet MVC Framework

Spring Framework: Module IV

Integration

- Remoting / Web Services
- EJB Integration
- Java Message Service (JMS)
- Java Management Extensions (JMX)
- Java Connector Architecture (JCA)
- Email
- Scheduling, Thead Pooling
- Unterstützung von dynamischen Sprachen (JRuby, Groovy, BeanShell)
- Annotations, Metadaten auf Sourcecode-Ebene

Referenzen

- IoC / DI
  - http://martinfowler.com/articles/injection.html
- Spring Framework
  - http://www.springframework.com
- Oates, Langer, Wille, Lueckow, Bachlmayr: „Spring & Hibernate, Eine praxisbezogene Einführung“, 2007 Carl Hanser Verlag München