

# Modele komponentowe SCA, OSGi, Distributed OSGi i OSGi Enterprise a Java EE

Jacek Laskowski

<http://www.JacekLaskowski.pl>

Wersja z 26 marzec 2010 o 15:50:32



IT Project Management

PHP

.NET & C#

Java

March 26th, 2010 in Poznan, Poland



# Ja...cek Laskowski

- Pasjonat Korporacyjnej Javy (**Java EE**) i okolic
- Założyciel i lider **Warszawa Java User Group**
- Członek grup rozwojowych **Apache OpenEJB**, **Apache Geronimo** i in.
- Bloger **Notatnika Projektanta Java EE**  
<http://www.JacekLaskowski.pl>
- Służbowo: **Specjalista** od oprogramowania **IBM WebSphere (BPM)** w IBM Polska





## Paweł Stawicki napisał...

*Ja bym chętnie zobaczył, jak się to wszystko "składa do kupy". Sporo tu elementów, o których mam co najwyżej blade pojęcie i ciekaw jestem, jak je wszystkie razem skonfigurować, połączyć i zbudować aplikację.*



Zacznijmy od początku.  
Króciutko



IT Project Management

PHP

.NET & C#

Java

March 26th, 2010 in Poznan, Poland

**Interfejs** to **zbiór metod**  
z określonym sposobem ich wykonania  
(**transport**)



**Usługa** to funkcjonalność  
dostępna za pomocą jednoznacznie  
zdefiniowanego **interfejsu**  
(w tym i transportu)



**Komponent** to **moduł** systemu  
udostępniający **usługi**  
bez odkrywania szczegółów  
ich implementacji  
**jednostka funkcjonalna/usługowa**



**Model komponentowy** to **zbiór zasad**  
określających  
ramy poprawnego funkcjonowania  
**komponentów**



# CORBA

Common Object Request Broker Architecture



IT Project Management

PHP

.NET & C#

Java

March 26th, 2010 in Poznan, Poland

# Nomenklatura i semantyka

Różnice między...

**model komponentowy** to **model aplikacyjny**?

A co ze **szkielet** (*framework*) i **rusztowanie**  
(*scaffolding*)

?



A wszystko zaczęło się od...?



IT Project Management

PHP

.NET & C#

Java

March 26th, 2010 in Poznan, Poland

```
package pl.jaceklaskowski.greeter;
```

```
public interface Greeter {  
    public String sayHelloTo(String who);  
}
```

```
package pl.jaceklaskowski.greeter.impl;
```

```
public class PolishGreeter implements Greeter {  
    @Override  
    public String sayHelloTo(String who) {  
        return "Witaj " + who;  
    }  
}
```



# Brawa dla człowieka-kompilatora!



IT Project Management

PHP

.NET & C#

Java

March 26th, 2010 in Poznan, Poland

# JavaBeans aka POJO

*JavaBeans technology is the **component architecture** for the Java 2 Platform, Standard Edition (J2SE). JavaBeans components (**beans**) are reusable software programs that you can develop and assemble easily to create sophisticated applications.*

Za <http://java.sun.com/javase/technologies/desktop/javabeans/index.jsp>



Później, znaczne skomplikowanie z...



# POJO + META-INF/ejb-jar.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<ejb-jar>
```

```
  <enterprise-beans>
```

```
    <entity>
```

```
    ...
```

```
  </entity>
```

```
</enterprise-beans>
```

```
</ejb-jar>
```



# Enterprise JavaBeans (EJB)

**Enterprise JavaBeans (EJB) technology is the *server-side component architecture* for Java Platform, Enterprise Edition (Java EE). EJB technology enables rapid and simplified development of distributed, transactional, secure and portable applications based on Java technology.**

Za <http://java.sun.com/products/ejb/>



Aby później uprościć?



# POJO + META-INF/beans.xml

```
<beans>
```

```
  <bean id="greeterService"
```

```
    class="pl.jaceklaskowski.greeter.impl.PolishGreeter">
```

```
</beans>
```



# Spring Framework

**Spring is**

*a layered Java/J2EE **application platform**.*

*Spring includes: The most complete lightweight container, providing centralized, automated configuration and wiring of your application objects. The container is non-invasive, capable of assembling a complex system from a set of **loosely-coupled components (POJOs)** in a consistent and transparent fashion.*

Za <http://www.springsource.org/about>



Ale nie tylko!  
POJO + adnotacje rulez :-)



# Java Enterprise Edition (Java EE)

*Java Platform, Enterprise Edition (Java EE) is the industry standard for **enterprise Java computing**.*

*The Java Platform, Enterprise Edition (Java EE) reduces the cost and complexity of developing **multitier, enterprise services**.*

Za <http://java.sun.com/javaee/> oraz  
Java Platform, Enterprise Edition (Java EE) Specification, v6



A w tle tli się coraz jaśniej...



IT Project Management

PHP

.NET & C#

Java

March 26th, 2010 in Poznan, Poland

# POJO + META-INF/MANIFEST.MF

Manifest-Version: 1.0

Bundle-ManifestVersion: 2

Bundle-Name: Greeter Service

Bundle-SymbolicName: pl.jaceklaskowski.greeter.Greeter

Bundle-Version: 1.0.0

Export-Package: pl.jaceklaskowski.greeter



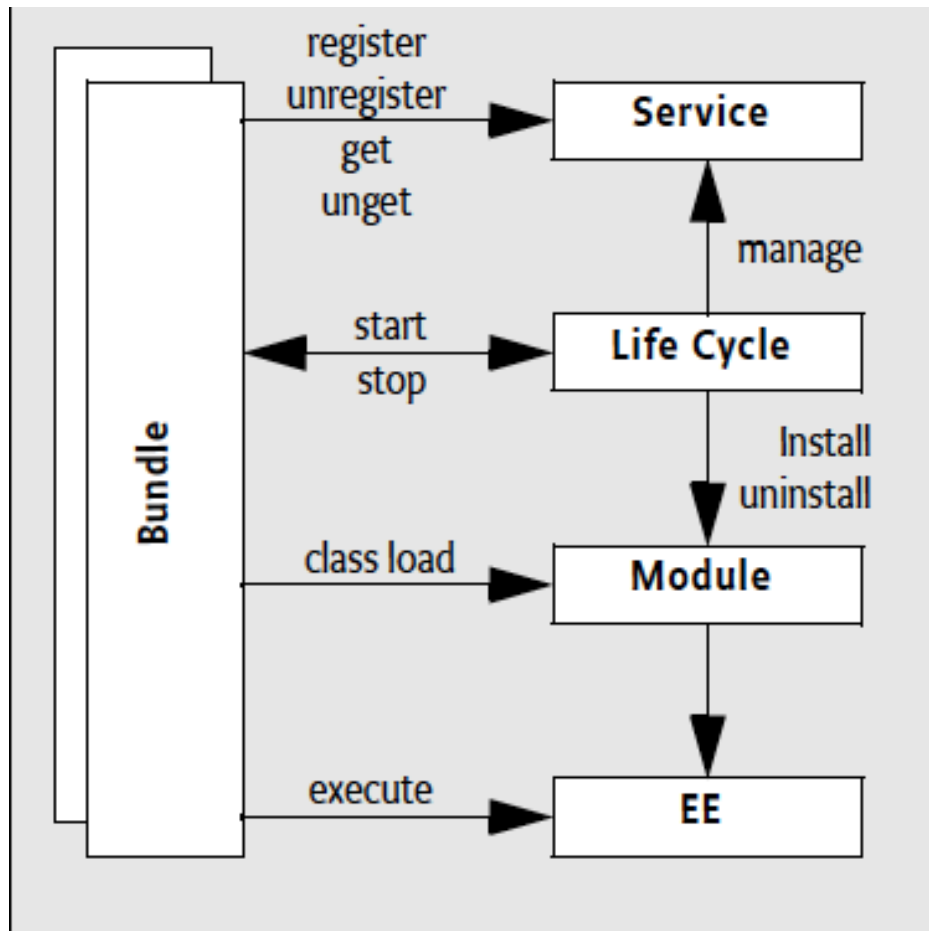
# OSGi

**OSGi** technology is the **dynamic module system for Java**. The OSGi Service Platform provides functionality to Java that makes Java the premier environment for software integration and thus for development.

Za <http://www.osgi.org/About/Technology>



# OSGi warstwami



- `registerService(String, Object, Dictionary)`
- `registerService(String [], Object, Dictionary)`



Za OSGi Service Platform Release 4 Version 4.2 Core Specification

IT Project Management

PHP

.NET & C#

Java

March 26th, 2010 in Poznan, Poland

Mając podstawy  
w postaci Java EE, Spring i OSGi  
zaczyna się prawdziwa,  
deklaratywna zabawa...



## META-INF/spring/simpleservice.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"  
  xsi:schemaLocation="http://www.springframework.org/schema/beans  
    http://www.springframework.org/schema/beans/spring-beans.xsd">  
  <bean name="simpleService"  
    class="org.springframework.osgi.samples.simpleservice.impl.MyServiceImpl" />  
</beans>
```

## META-INF/spring/simpleservice-osgi.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:osgi="http://www.springframework.org/schema/osgi"  
  xsi:schemaLocation="http://www.springframework.org/schema/beans  
    http://www.springframework.org/schema/beans/spring-beans.xsd  
    http://www.springframework.org/schema/osgi  
    http://www.springframework.org/schema/osgi/spring-osgi.xsd">  
  <osgi:service id="simpleServiceOsgi" ref="simpleService"  
    interface="org.springframework.osgi.samples.simpleservice.MyService" />  
</beans>
```



# Spring Dynamic Modules for OSGi(tm) Service Platforms

=

Spring Framework + OSGi



# OSGI-INF/blueprint/blueprint.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">

  <reference id="greeterService"
    interface="org.apache.aries.tutorials.blueprint.greeter.api.GreeterMessageService"
    timeout="10000" availability="mandatory" />

  <bean id="clientBean"
    class="org.apache.aries.tutorials.blueprint.greeter.client.GreeterBlueprintClient"
    activation="eager" init-method="doRequests">

    <argument value="Blueprint Client" />
    <property name="greeterMessageService" ref="greeterService" />

  </bean>

</blueprint>
```



# OSGi Blueprint Container

**Blueprint Container** specification defines a **dependency injection framework**, specifically for **OSGi bundles**, that understands the unique dynamic nature of services.

This Blueprint Container specification is derived from the Spring Dynamic Modules project.

Za 121. Blueprint Container Specification w OSGi Service Platform  
Release 4 Version 4.2 Enterprise Specification



Wychodzimy poza ramy pojedynczej JVM...  
Gotowi?



```
public class Activator implements BundleActivator {  
    private ServiceRegistration registration;  
  
    public void start(BundleContext bc) throws Exception {  
        Dictionary props = new Hashtable();  
  
        props.put("service.exported.interfaces", "*");  
        props.put("service.exported.configs",  
                "org.apache.cxf.ws");  
        props.put("org.apache.cxf.ws.address",  
                "http://localhost:9090/greeter");  
  
        registration =  
            bc.registerService(GreeterService.class.getName(),  
                               new GreeterServiceImpl(), props);  
    }  
  
    public void stop(BundleContext bc) throws Exception {  
        registration.unregister();  
    }  
}
```



# Distributed OSGi

to potoczna nazwa dla

**Remote Services** (OSGi Compendium spec)

oraz

**Remote Service Admin Service** (OSGi  
Enterprise spec)



**Enterprise OSGi**  
to zbiór specyfikacji w  
**OSGi Service Platform Release 4 Version 4.2**  
**Enterprise Specification**  
dla

Blueprint Container, Remote Service Admin  
Service, JTA, JMX, JDBC, JNDI, JPA, Web i SCA



Ktoś wspomniał o SCA?! W końcu!



# Service Component Architecture (SCA)

**Service Component Architecture (SCA)** is a **set of specifications** which describe a **model** for building applications and systems using a Service-Oriented Architecture. SCA extends and complements prior approaches to implementing services, and SCA builds on open standards such as Web services.

Za <http://www.osoa.org/display/Main/Service+Component+Architecture+Home>



# POJO + META-INF/\*.composite

```
<?xml version="1.0" encoding="ASCII"?>
<!-- MyValueComposite_1 example -->
<composite xmlns="http://www.oxa.org/xmlns/sca/1.0"
           targetNamespace="http://foo.com"
           name="MyValueComposite" >

  <service name="MyValueService" promote="MyValueServiceComponent"/>

  <component name="MyValueServiceComponent">
    <implementation.java class="services.myvalue.MyValueServiceImpl"/>
    <property name="currency">EURO</property>
    <reference name="customerService"/>
    <reference name="stockQuoteService"/>
  </component>

  <reference name="CustomerService"
            promote="MyValueServiceComponent/customerService"/>

  <reference name="StockQuoteService"
            promote="MyValueServiceComponent/stockQuoteService"/>

</composite>
```



# SCA, OSGi, Spring i Java EE

- W OSGi Enterprise – SCA Configuration Type Specification
- W SCA – implementation.\*:
  - \*.jee, \*.web i \*.ejb dla artefaktów Java EE
  - \*.osgi, \*.java i \*.spring dla OSGi, Java i Spring



Więcej kodu zamiast gadania?!  
Muszę Was rozczarować, więcej nie będzie,  
bo nie ma być!



# Podsumow(yw)anie



IT Project Management

PHP

.NET & C#

Java

March 26th, 2010 in Poznan, Poland

# Programowanie deklaratywne

- POJO
- Adnotacje
- Pliki konfiguracyjne (często opcjonalne)
  - META-INF/beans.xml,  
WEB-INF/applicationContext.xml – Spring
  - META-INF/MANIFEST.MF – OSGi
  - META-INF/\*.composite – SCA
  - META-INF/ejb-jar.xml – EJB (Java EE)



# Środowiska uruchomieniowe

- OSGi – Apache Felix, Eclipse Equinox
- Distributed OSGi (d-OSGi) – Apache CXF, Eclipse ECF
- Enterprise OSGi (ent-OSGi) – Apache Aries, Eclipse Virgo (Spring-DM)
- SCA – Apache Tuscany, Eclipse SCA Tools



# Dalsza lektura

- OSGi Service Platform Release 4 Version 4.2  
Core Specification  
<http://www.osgi.org/Download/File?url=/download/r4v42/r4.core.pdf>
- OSGi Service Platform Release 4 Version 4.2  
Compendium Specification  
<http://www.osgi.org/Download/File?url=/download/r4v42/r4.cmpn.pdf>
- OSGi Service Platform Release 4 Version 4.2  
Enterprise Specification  
<http://www.osgi.org/Download/File?url=/download/r4v42/r4.enterprise.pdf>
- Service Component Architecture Specifications  
<http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications>



# Modele komponentowe SCA, OSGi, Distributed OSGi i OSGi Enterprise a Java EE

Jacek Laskowski

<http://www.JacekLaskowski.pl>



IT Project Management

PHP

.NET & C#

Java

March 26th, 2010 in Poznan, Poland