About

OSGi and Java Enterprise consultant

Java User Group Milano founder and coordinator

Apache Felix committer

http://www.diotalevi.com
filippo@diotalevi.com
Summary

OSGi

Spring Dynamic Modules for OSGi

What’s next
OSGi

A dynamic module system for Java

(Demo)
Module System

Clear boundaries

Dependencies

Metadata

Lifecycle
Lifecycle of bundle

- Install
  - Installed
  - Resolved
    - Start
    - Stop
  - Active
Service System

Service Oriented Architecture in the JVM
Service System

Service Registry

Register Service

Bundle Provider

Retrieve Service

Bundle Consumer
What’s an OSGi Application?

A collection of bundles interacting via Service interfaces

Bundles developed and installed independently

Bundle and services can appear and disappear
What's behind the demo

Service Registry

Bundle Host
*Provide the Frame and define the Shape interface*

Listen for and retrieve *Shapes*

Register Triangle

Bundle Circle
*Provide the Circle implementation*

Register Circle

Bundle Triangle
*Provide the Triangle implementation*
Why Spring DM?

OSGi in **NOT** about user interfaces

OSGi is about building modular and service oriented applications

Enterprise applications need modularity and SOA
Spring Dynamic Modules

Part of the Spring open source portfolio

Formerly known as Spring-OSGi

Works with every OSGi container

Allows you to implement Spring applications on top of any OSGi framework
Spring DM idea

Spring + Spring DM

OSGi Container

Java Virtual Machine
Bundles & Spring

One application context per bundle

When a bundle is started, its application context is discovered and instantiated

Bundles can export Spring beans as OSGi services

Bundles can import OSGi services
Beans as OSGi Services

<beans>
    <bean name="customerService"
          class="demo.customerservice.impl.CustomerServiceImpl"/>

    <osgi:service id="customerServiceOsgi" ref="customerService"
                  interface="demo.customerservice.CustomerService"/>
</beans>
Importing OSGi services

<beans>

  <osgi:reference id="customerServiceOsgi"
      interface="demo.osgi.customerservice.CustomerService"/>

  <bean id="orderService"
      class="demo.osgi.orderservice.internal.OrderServiceImpl">
    <property name="customerService">
      <reflocal="customerServiceOsgi"/>
    </property>
  </bean>

</beans>
Bigger picture

Application Context

Customer Service

Service Registry

Exposed Customer Service

Retrieve Customer Service

Application Context

Order Service

use

Customer Service Proxy
Result

(Almost) no dependencies on OSGi APIs

Everything is a POJO (Spring philosophy)

Improved application modularity
Going forward

Standardization: OSGi Blueprint Service

Enterprise OSGi support in Spring DM
Full support for Web Applications
and Java EE components

Integration with OSGi Remote-Services
Inter-JVM services
Questions?
Images used:

http://www.flickr.com/photos/estringita/2722410302/
http://www.sxc.hu/photo/1124847
http://www.sxc.hu/photo/1222662
http://www.sxc.hu/photo/1038332
http://www.sxc.hu/photo/1039992
http://www.sxc.hu/photo/994582