Modularidade com Java Module System & OSGi

Vinicius Senger

vinicius@globalcode.com.br

vsenger.blogspot.com

Agenda

- Novos problemas
- OSGi Framework
- Java Module System
- Modularidade com Java SE e EE
- Conclusão

Objetivos

Explorar a situação atual de técnicas para componentização e modularidade de aplicativos Java de grande porte.

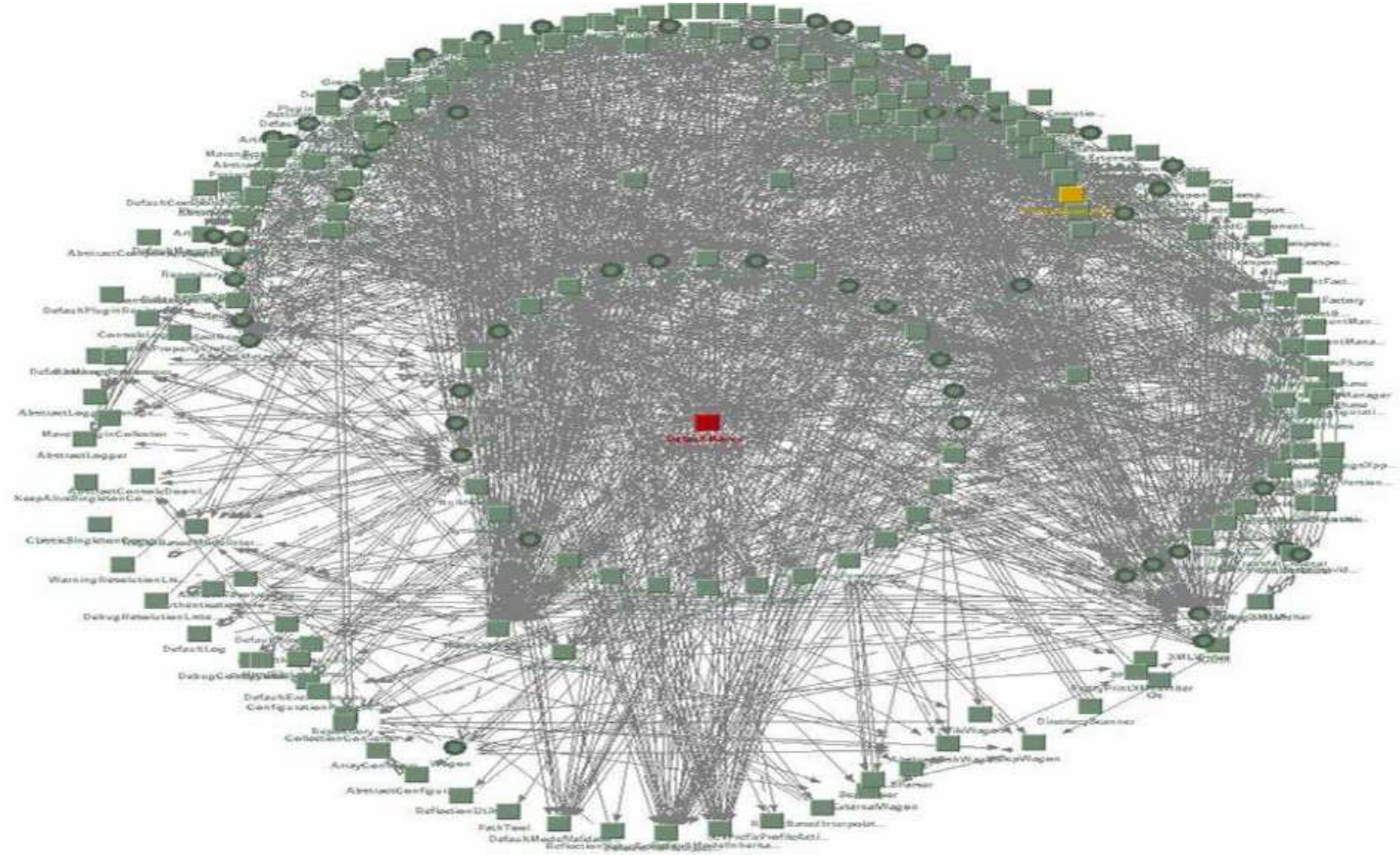
Agenda

- Novos problemas
- OSGi Framework
- Java Module System
- Modularidade com Java SE e EE
- Conclusão

Novos problemas

- Orientação a objetos **não** significa **componentização**;
- Normalmente trabalhamos com apenas **um** classpath por máquina virtual;
- JARs são frequentemente atualizados;
- Alto acoplamento de outros JARs;

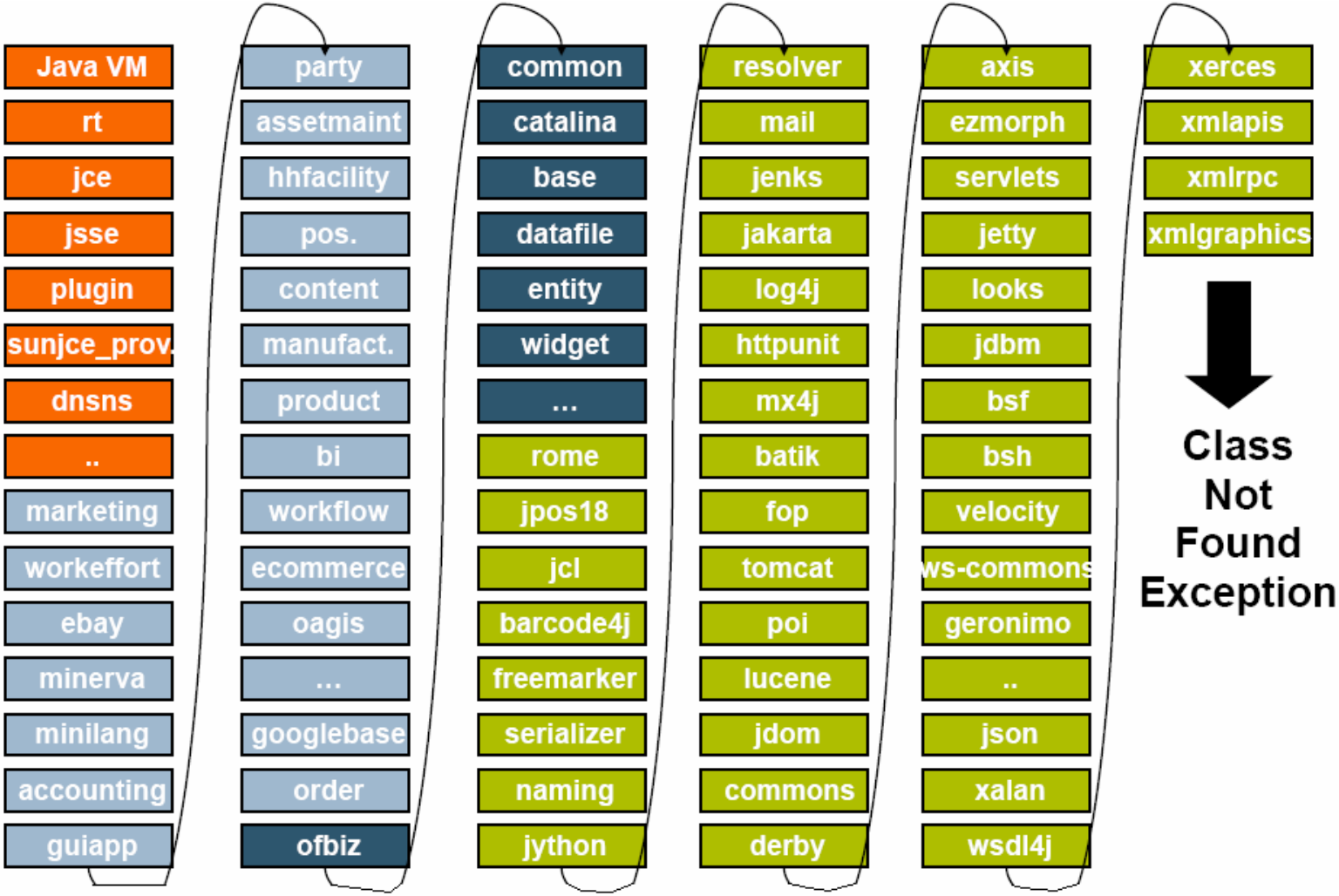
O mundo OO / JAR



Novos problemas

- O sistema de classloader do Java não suporta versionamento, na prática:
  - Como podemos ter duas versões de um mesmo EJB rodando na mesma máquina virtual?
- public / protected / default / private não são suficientes;
- É preciso um public “si pero no mucho”;

Carregamento de classes



Novos problemas

- SOA, injeção de dependência, factories e frameworks tentam minimizar parte destes problemas;
- OO não é suficiente;
- JAR não é suficiente;
- public / protected / default / private não são suficientes;
- ClassLoader não é suficiente;

Agenda

- Novos problemas
- OSGi Framework
- Java Module System
- Modularidade com Java SE e EE
- Conclusão

OSGi Framework

- OSGi é um padrão de modelo de componentização para arquiteturas modulares;
- O padrão é mantido pela OSGi Alliance;
- Define uma série de serviços: log, config, preferences, event, deployment, HTTP, etc.;
- JSR-291 = Dynamic Component Support for JavaTM SE

OSGi Framework

- É uma uma infraestrutura para tornar aplicativos mais modulares:
  - Permite múltiplas versões de uma mesma classe dentro da mesma JVM;
  - Sistema de exportar pacotes mais robusto;
  - Controle de ciclo de vida de serviços;
  - Controle rigoroso nas dependências;
  - Tracking de serviços;

OSGi Framework

*"The OSGi specifications are so widely applicable because the platform is a **small layer** that allows **multiple Java**™ based **components** to efficiently **cooperate** in a **single** Java Virtual Machine (**JVM**)."*

*Texto retirado do site www.osgi.org

OSGi Framework

- OSGi trabalha com conceito de *bundle* = *aplicativo ou parte dele*;

- Um bundle seria o equivalente a um JAR, porém com informações manifest adicionais:

  - Identidade;
  - versão;
  - Dependências;
  - Exposições / publicações;

OSGi Framework

> DEMO: O mundo OSGi...

OSGi Framework

"*We are looking at OSGi the same way many other vendors are looking at it, as an* ***enabling technology*** *for* ***dividing*** *up a* ***large software*** *project and deploying the results.*"

Eric Newcomer – fundador IONA

* Trecho de entrevista Infoq.com

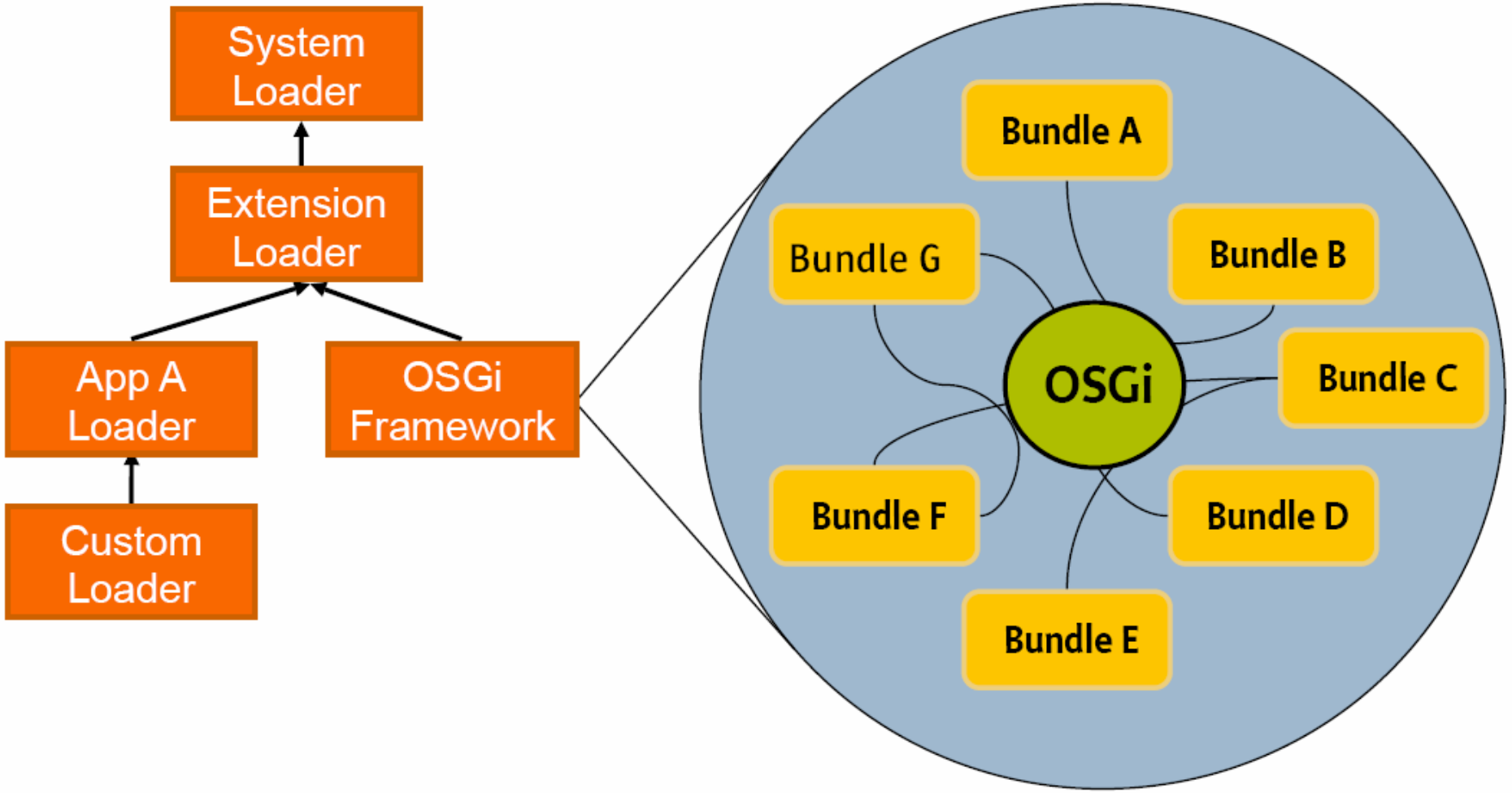
OSGi Framework

- Bundles podem ser instalados e gerenciados remotamente;
- OSGi favorece arquiteturas orientadas a serviços;
- Eclipse, WebSphere, Jboss, Jonas, BEA e Glassfish adotaram OSGi;

OSGi Framework

- Não é comum encontrarmos OSGi puro em cenários de negócio;
- OSGi Enterprise Expert Group foi criado para suportar as necessidades de vendors e desenvolvedores Java EE;
- Futuramente devemos poder empacotar componentes EJB e beans Web como serviços OSGi;

OSGi Framework



OSGi Framework

> Sua JVM pode sustentar mais de um branch CVS?

> DEMO: O mundo das versões...

OSGi Framework

"*This is similar to the **service-oriented architecture** made popular with **web services**. The key **difference** between **web services** and **OSGi services** is that web services **always** require some **transport layer**, which makes it **thousands times slower than OSGi** services that use direct method invocations.* "

*Retirado do site www.osgi.org

OSGi Framework

- Criando um serviço OSGi:
  - Definimos uma interface, que será exposta:

```
package sendmail.service;
public interface MailService {
  public void enviar(Mail mail);
}
```

  - Implementamos a interface em uma classe que não será exposta;

```
package sendmail.impl;
public class MailServiceImpl implements MailService {
  public void enviar(Mail mail) { ... }
}
```

OSGi Framework

> Devemos criar um ativador do serviço:

```
public class Activator implements BundleActivator {
  private MailService service;
  private ServiceTracker mailServiceTracker;
  public void start(BundleContext context) throws Exception {
    service = new MailServiceImpl();
    context.registerService(MailService.class.getName(),
       service, new Hashtable());
    mailServiceTracker = new ServiceTracker(
      context, MailService.class.getName(), null);
    mailServiceTracker.open();
  }
  public void stop(BundleContext context) throws Exception {
    helloServiceTracker.close();
    helloServiceTracker = null;
    service = null;
  }
}
```

OSGi Framework

> Manifest.mf:

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: SendMail Plug-in
Bundle-SymbolicName: SendMail
Bundle-Version: 1.0.0
Bundle-Activator: sendmail.Activator
Import-Package: org.osgi.framework;version="1.3.0",
 org.osgi.util.tracker;version="1.3.1"
Export-Package: sendmail.service
```

OSGi Framework

> Demo: rodando OSGi com Eclipse

Agenda

- Novos problemas
- OSGi Framework
- Java Module System
- Modularidade com Java SE e EE
- Conclusão

Java Module System

- Padrão JCP para resolver problemas semelhantes as propostas do OSGi;
- Bundle = JAM (Java Module);
- JSR 277: Java Module System:

  "The R3 version of the Open Services Gateway Initiative (OSGi) specification defines a framework that enables the deployment of service-oriented applications (called bundles). However, the framework only supports package dependency based on the minimum version of a specification, and there is no support for exact version or version range. "

Java Module System

- Hibernate, Struts, JSF, Unified Process, Spring, Seam, OJB, AspectJ, WebWork, Agile...
- OSGi & Java Module System: a nova guerra?

Java Module System

> JDK 7 terá o Java Module System interoperando com OSGi;

"*We're announcing the availability of a new specification that defines the interoperability between the Java module system for SE 7 and OSGi bundles,"*

What this means is in JDK 7, developers who create applications that use OSGi bundles will be able to run them unmodified on JDK 7. “

Danny Coward – Sun Microsystems

Java Module System

"I'm wondering why they needed to do the module support and not just go with OSGi support. But as long as the module support fully supports OSGi, that's OK,"

Anne Thomas Manes – Burton Group

Agenda

- Novos problemas
- OSGi Framework
- Java Module System
- Modularidade com Java SE e EE
- Conclusão

Modularidade com SE e EE

- Devemos evitar um número excessivo de classes por JAR;
- Evitar o uso de WEB-INF/classes;
- Conheça bem o sistema de classloader do seu application server;
- Usando JBoss, recomendamos o uso de SAR;

Modularidade com SE e EE

- Evitar o uso de Class.forName;
- Evitar a construção de Classloaders;
- Adote uma infraestrutura moderna: seam / jboss microkernel, spring ou glassfish 3;
- As soluções definitivas chegarão com Java 7 (dolphin);

Modularidade com SE e EE

- Ao adotar OSGi:
  - Comece transformando seu aplicativo em um bundle;
  - Isole serviços em outros bundles;
  - Desenhe as dependências de classes;
  - Utilize um modelo de programação mais alto nível:
    - Apache iPOJO
    - Spring DM
    - Jboss 5

Agenda

- Novos problemas
- OSGi Framework
- Java Module System
- Modularidade com Java SE e EE
- Conclusão

Conclusão

- OSGi é madura e robusta, mas adotar no baixo nível pode não ser adequado para cenários de negócio simples;
- As especificações Java permitirão múltiplos sistemas de módulos;
- OSGi = componentização + arquitetura orientada a serviços;

Links

- http://www.osgi.org/About/Technology
- http://www.eclipse.org/equinox/
- http://www.javaworld.com/javaworld/jw-05-2008/jw-05-rightsize.html?page=1
- http://www.knopflerfish.org